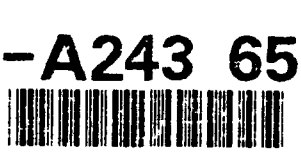


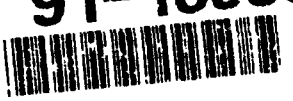
AFIT/GE/ENG/91D-06

ELECTRICALLY ERASABLE PROGRAMMABLE INTEGRATED CIRCUITS FOR REPLACEMENT OF OBSOLETE TTL LOGIC

THESIS

Joseph V. Breen, GS-12

AFIT/GE/ENG/91D



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1991	Master's Thesis

4. TITLE AND SUBTITLE
ELECTRICALLY ERASABLE PROGRAMMABLE INTEGRATED CIRCUITS FOR REPLACEMENT OF OBSOLETE TTL LOGIC

5. FUNDING NUMBERS

6. AUTHOR(S)
Joseph V. Breen

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/91D-06

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
SM-ALC/TIEFD (Eric Campbell)
McClellan AFB, CA 95652

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Two microcircuits with electrically erasable programmable logic arrays, which use Fowler-Nordheim (F-N) tunneling for both programming and erasing, were designed to demonstrate the use of programmable logic for obsolete TTL logic replacement. Each microcircuit was fabricated in the Orbit 2-micron double-poly low noise analog CMOS process through MOSIS. Software to generate VHDL structural models from a pin list was developed and the logic of both designs was verified by simulation using the Zycad VHDL simulator.

The first microcircuit included a simple programmable logic circuit and test cells that allowed measurement of the programming characteristics of the floating gate transistors. Test results on this microcircuit show a wide variance of programmability even between similar transistors on the same die. Some evidence of F-N tunneling from the control gate to the floating gate was also noted.

The second microcircuit was designed to emulate the logic of a group of combinational TTL circuits. VHDL simulation was successful for each emulation. Device testing provided limited results due to a design error; however, successful emulation of two sections of a 7400 quad-NAND circuit was achieved.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Programmable Logic, Obsolete Integrated Circuit Replacement, MOSIS, VHDL, Logic Circuits, Transistor Transistor Logic	239
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

AFIT/GE/ENG/91D-06

ELECTRICALLY ERASABLE PROGRAMMABLE INTEGRATED CIRCUITS FOR REPLACEMENT OF OBSOLETE TTL LOGIC

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Joseph V. Breen, B.S.

December 1991

Acknowledgements

First, I wish to thank my wife Beverly for her understanding and support. During this research there were certain periods, most notably as deadlines for design submission for IC fabrication approached, that it seemed like there were not enough hours in the day to accomplish the required work. During those times her support and encouragement were most welcome.

Many people helped me to perform the research documented in this thesis. In particular, I would like to thank my thesis advisor, Capt. Mark Mehalic for his encouragement and ideas. I would also like to thank the other members of my thesis committee, Major Bill Hobart and Capt. Keith Jones for their assistance.

Further thanks go to the people at Sacramento ALC, particularly Bob Price, Ted Glum, and Eric Campbell, who allowed me to come to AFIT, and to Ellis Anderson at the Scientist and Engineer Civilian Career Enhancement Program (SECCEP) office who encouraged me to apply.

Joseph V. Breen

Table of Contents

List of Figures

List of Tables

Abstract

Two microcircuits with electrically erasable programmable logic arrays, which use Fowler-Nordheim (F-N) tunneling for both programming and erasing, were designed to demonstrate the use of programmable logic for obsolete TTL logic replacement. Each microcircuit was fabricated in the Orbit 2-micron double-poly low noise analog CMOS process through MOSIS. Software to generate VHDL structural models from a pin list was developed and the logic of both designs was verified by simulation using the Zycad VHDL simulator.

The first microcircuit included a simple programmable logic circuit and test cells that allowed measurement of the programming characteristics of the floating gate transistors. Test results on this microcircuit show a wide variance of programmability even between similar transistors on the same die. Some evidence of F-N tunneling from the control gate to the floating gate was also noted.

The second microcircuit was designed to emulate the logic of a group of combinational TTL circuits. VHDL simulation was successful for each emulation. Device testing provided limited results due to a design error; however, successful emulation of two sections of a 7400 quad-NAND circuit was achieved.

ELECTRICALLY ERASABLE PROGRAMMABLE INTEGRATED CIRCUITS FOR REPLACEMENT OF OBSOLETE TTL LOGIC

I. Introduction

Background

Since the late 1960's Integrated Circuits (ICs) have been used extensively in weapons systems designs. Advances in IC manufacturing technology make older IC designs obsolete. Since the more recently designed ICs are faster, have better drive characteristics, use less power, and implement more functionality in a given package size, designers use the newer ICs in new designs. This leads to a decrease in demand for older types which reduces the profitability of manufacturing these types. Eventually, manufacturers will discontinue manufacturing the components with reduced demand.

Weapons systems have an extended lifetime when compared to commercial systems, which creates a need for replacement ICs in weapons systems after the large volume requirements of commercial systems have ceased and the manufacturers have ceased production. This need is often met by purchasing enough of any given type of component to meet the expected demand for that component for the projected lifetime of the

system in which it is used. Shortages of components can still occur when demand for a given component exceeds the expected demand, components are used to manufacture replacement subassemblies, or a component type is discontinued without sufficient notice to allow lifetime purchases.

Many of the devices used from 1968 to the present belong to the Transistor-Transistor Logic (TTL) family. Sub-groups of the TTL family are Standard TTL (referred to as TTL, which is also the generic description of each sub-group), Low Power TTL (LTTL), High Power TTL (HTTL), Schottky TTL (STTL), Low Power Schottky TTL (LSTTL), Advanced Schottky TTL (ASTTL), Advanced Low Power Schottky TTL (ALSTTL), and some less common variants. All members of the TTL family are classified as bipolar devices because bipolar transistors are used throughout the devices. Certain members of the TTL family of ICs are becoming obsolete. Part types which are presently being affected are members of the Standard TTL, LTTL, and HTTL sub-groups.

Several methods for providing replacements for obsolete ICs have been proposed. The use of substitute parts is appropriate and cost effective when such substitutes are available. Re-opening the manufacturing lines and making new parts using the original designs is possible when the original design is available, but is an expensive option. Device arrays (semiconductor devices which consist of an

array of transistors, resistors, capacitors and diodes which are interconnected by the metallization layers in a final manufacturing step) emulate obsolete devices at an intermediate cost. Finally, the use of components which are specifically designed to be programmed before use, and which can emulate many different parts, may be the least expensive method of providing replacements for unavailable obsolete IC types which have no suitable substitutes.

An ideal replacement integrated circuit would perform identically to the IC being replaced in all circuits in which the original IC is used. Ideally, all circuits in which the original IC is used should have been designed such that they rely only on the published specifications of the IC and any IC meeting the parametric specifications of the original should perform correctly. In practice, a circuit may rely on characteristics of the original IC which are not strictly specified. One extreme example might be a circuit which relies on the fact that a given IC sources a certain amount of current at a given input. While the amount of current which flows out of an input pin when it is held low may be specified as typical and maximum values, a minimum may have been assumed such that a replacement IC which meets all of the specifications of the original IC but sources much less current at the input may not function correctly in the circuit. Examples of other characteristics of an IC

which may not be specified but which may be critical to correct operation in certain circuits include output rise time (where a maximum is specified but no minimum is specified), output drive capability, and anomalous outputs which may occur under certain conditions which a given circuit may require for correct operation. This type of dependence on non-specification values (or better than specification values) may occur when a design engineer builds a prototype circuit which functions correctly even though an analysis of the circuit operation using worst case values for the specified parameters may show that proper operation is not guaranteed by the specifications. Some circuits which depend on these unspecified characteristics, or require performance characteristics which significantly exceed the specifications of the devices used, may operate properly only with devices from an individual vendor or only with devices hand-picked to operate in the circuit. Thus the problem of developing a replacement IC which will function correctly in almost all circuits in which the original IC was used is more than simply meeting all of the published specifications of the original part.

Problem

A programmable device which can replace many different TTL devices is needed. The device should be capable of emulating the logic functions of the chips to be replaced as

well as meeting the published timing, logic level, drive, and input source specifications of the ICs to be replaced. While the device may be fabricated in a CMOS technology, it must not exhibit latchup or other destructive modes of operation when subjected to conditions which might be found in actual operation.

Scope

The scope of this project was limited to developing a replacement demonstration IC which is electrically erasable and can be programmed to emulate certain TTL components. The IC developed during this project was packaged in a standard 40 pin package, and is therefore not suitable for actual replacement use.

Methodology

Two microcircuits were developed. The first microcircuit was designed to demonstrate programmable floating gate transistors and a rudimentary programmable logic circuit. This circuit includes programmable test transistors which were used to determine the programming and erasing characteristics of floating gate transistors manufactured in the MOSIS Low Noise Analog (LNA) process. These transistors use Fowler-Nordheim tunneling for charge transfer to and from the floating gates.

The second microcircuit has a complex programmable logic array which is capable of emulating several

combinational TTL logic components. Program patterns were developed for the second microcircuit to demonstrate TTL emulation.

In both cases, VHDL models were used to demonstrate correct design and to test program patterns.

Presentation

A review of the literature on obsolete integrated circuit replacement options, Fowler-Nordheim tunneling, and floating gate transistors is presented in Chapter II. A description of the design, VHDL model, test apparatus, and test results for the first test microcircuit are presented in Chapter III. Chapter IV contains the description of the design, VHDL model, test apparatus, and test results for the second test microcircuit. A summary of the work performed, conclusions reached, recommendations for further research, and some lessons learned during the research effort are presented in Chapter V. Additional information, including software programs developed during the course of the research, complete schematic diagrams of each of the microcircuits, and the VHDL models for each of the microcircuits is presented in the appendices.

II. Literature Review

Integrated Circuit Replacement Options.

Modern weapons systems have a long life compared to systems of similar complexity used in the private sector. The weapons system manager must make repair components available during the extended lifetime of the system, and, if the manufacture of any higher assembly that incorporates the component becomes necessary, he must also ensure that small production quantities of the components are available. One type of repair component that presents special problems in weapons system support is the Integrated Circuit (IC). Of the 37,000 IC types stocked by the government, it is estimated that 24,000 bipolar IC types will go out of production in the next several years [1].

One reason that integrated circuits are more difficult to support than many other types of components is that technology obsolescence and low demand for older part types lead to the original manufacturer discontinuing the manufacture of these types [2]. The commercial market ends up concentrating on the newest, most profitable technologies at the expense of the users of older, less profitable lines. Government agencies can no longer procure entire families of parts, such as the Diode-Transistor Logic family, from the original manufacturers. Suggested methods of obtaining presently unavailable (or soon to become unavailable) IC

types or suitable replacements include: make lifetime buys, transfer the obsolete technologies to companies that would specialize in supporting obsolete IC types, substitute modern types for obsolete types, reverse engineer and remanufacture obsolete ICs, develop device arrays that can emulate obsolete ICs, and emulate obsolete types with programmable components.

Lifetime Buys. The lifetime buy is the traditional method for supporting obsolescent IC types [3]. When a manufacturer notifies the DOD that a particular component type is about to be discontinued, an estimation is made of the quantity of components required to support systems that incorporate the component for the predicted lifetimes of the systems and the estimated number of components are purchased as spare parts. Difficulties with this method include accurately estimating the number of components required and, in some cases, insufficient lead time to develop a plan for supporting the IC.

Technology Transfer. The quantity of obsolete components required to support weapons systems is small in comparison to commercial production quantities; however, sufficient demand may exist to support companies formed with the specific purpose of providing such obsolete components. One such company, Lansdale Transistor and Electronics, has purchased the rights to and the production line for the

Sylvania Universal High Level Logic family of obsolete ICs [4]. Arnold Stensrude, director of marketing for military IC products at Motorola, has suggested that the government "adopt a plan and [periodically] transfer older military integrated circuit technologies to the after-market manufacturer [2]."

Substitution. Substituting similar types of available ICs for no longer available types is a cost effective alternative to exact replacements. For several reasons, this method is not always applicable. First, vendor specific chips and the use of increasingly complex chips in military systems may mean that no suitable substitute IC exists [2]. Second, determining which characteristics of an IC are critical to proper operation in a circuit may be difficult or impossible. King [5] states that having an application including several dozen other ICs and their interconnects would aid in the analysis process. In addition, the unavailable IC may be used in many different circuits, some of which may rely on "undocumented or unspecified parameters" possibly making the choice of a substitute IC dependent on the target application [1].

Automated analysis techniques have been suggested for applying a specific substitution technique. The lack of data bases of designs in machine readable formats may hinder such an effort. Other aids to substitution of IC types

include pin scramblers and small ceramic carriers mounting small outline IC packages [5].

Reverse Engineering and Remanufacturing. The obsolete IC procurement problem is made worse by the lack of adequate parts documentation. For ICs added to the inventory in the future, it has been suggested that each new part be fully specified in a hardware description language such as the VHSIC Hardware Description Language (VHDL) [3]. Since such data is not typically available on currently obsolete or nearly obsolete ICs, reverse engineering (the process of analyzing actual devices for topological, physical, and electrical parameters) may be necessary. Attempts to copy devices directly from the device itself have met with mixed success. In one case, eight device types were copied, but only three functioned properly [1].

Emulation Using Device Arrays. Emulation using device arrays uses one moderately complex IC with customized final metallization layers to replace many different IC types. The IC is made up of many uncommitted devices and device arrays, such as resistors, capacitors, transistors, and diodes, on the silicon chip. One example is the Honeywell Bipolar Device Array (BDA), developed as a replacement for many obsolete components in the Diode Transistor Logic (DTL) family. The BDA design includes provisions for obtaining the desired input current, drive capability, and delay

characteristics. Eight different components have been successfully emulated using the BDA and a design manual is available to support the design of devices using the BDA [6].

While the use of device arrays promises to replace almost all difficult or special cases for which no standard circuit in a new technology exists, the cost is relatively high. Prices in the tens of thousands of dollars per emulation are being quoted [5].

Emulation With Programmable Components. Programmable components are components that have certain essential characteristics selected or programmed before use. Several types of programmable devices may be especially useful for obsolete IC replacement. One such type, the Erasable Programmable Logic Device (EPLD), was used by NASA to replace logic on a printed circuit board. In the specific case reported, a printed circuit board redesign was necessary; however, the authors stated "They also offer a potential solution to the problem of discontinued devices since the same programmable device can replace many different discrete devices" [7].

Fowler-Nordheim Tunneling

Simplified Theory. Electrons in polysilicon are usually prevented from entering SiO_2 by an energy barrier of approximately 3.2 eV; however, electrons have a small probability of tunneling a short distance into the SiO_2. At room tem-

perature the kinetic energy of the electrons will allow a tunneling distance of approximately 50Å. Providing the electric field within the SiO_2 overcomes the energy barrier, the tunneling electrons will not return to the polysilicon but will be carried by the electric field, causing a current to flow. The field intensity required to create a potential difference of 3.2V across a 50Å distance is $3.2V/5 \times 10^{-7}$ cm $= 6.4 \times 10^6$ V/cm. Increasing the electric field above this level will cause larger currents to flow because more electrons will tunnel into the SiO_2 the correspondingly shorter distance required to exceed the 3.2V potential barrier [8].

Tunneling Current. The tunneling current density can be represented as:

$$J=\frac{1.54 \times 10^{-6} E^2}{\Phi_B} \exp\left[\frac{-6.83 \times 10^7 \Phi_B^{3/2}(m^*/m)^{1/2}}{E}\right] \qquad (1)$$

where J is the current density in A/cm^2; E is the electric field in V/cm; m^*/m is the average effective mass of electrons in the oxide forbidden band; and Φ_B is the barrier energy (3.2eV) [9]. The average effective mass ratio (m^*/m) has been variously reported as 0.47 [9], 0.42 [10], and 0.50 [11].

Lenzlinger and Snow [10] experimentally determined that tunneling current densities in SiO^2 were an order of magnitude lower than predicted by (1) when measured after the

experimental devices had been subjected to an initial tunneling current of 10^{-10} A/cm^2 for two hours, during which time the current density decreased by an order of magnitude. The reduction in tunneling current density was attributed to electron trapping in the oxide.

Kolodny et. al. [12] also experimentally determined the Fowler-Nordheim tunneling current density. Their experimental devices were pre-conditioned by passing 0.1 C/cm^2 of charge has through the oxide. The tunneling current density for oxide thicknesses of 100-150 Å was found to be:

$$J_{tun} = \alpha E_{tun}^2 \exp(-\beta/E_{tun}) \qquad (2)$$

where $\alpha = 1.88 \times 10^{-6}$ A/V^2, $\beta = 2.55 \times 10^8$ V/cm, and E is the electric field in the oxide. Calculating the constants in (2), α and β, from (1) and reducing the magnitude by a factor of ten to allow for the observed reduction of current flow over time, yields $\alpha = 4.8 \times 10^{-6}$ A/V^2 and $\beta = 2.36 \times 10^8$ V/cm. While the two values for β agree well, the value for α calculated from (1) is over twice the adjusted value given by (2). From an initial value of 10^{-10} A, and assuming a linear decrease in current over the two hour period, the total charge transfer would be 6.5×10^{-7} C for the devices tested by Lenzlinger and Snow. Even if the current were assumed to be constant at 10^{-10} A for the two hour period, the total charge transferred would be 7.2×10^{-7} C. The amount of charge that was passed through the experi-

mental samples before measuring the tunneling current by Kolodny et. al. is five orders of magnitude larger than the amount of charge passed through the experimental samples used by Lenzlinger and Snow. This large difference in total charge transported may account for the higher tunneling current density reported by Lenzlinger and Snow.

Floating Gate Transistors

A floating gate transistor is an insulated-gate field effect transistor (FET) that has a gate, usually made of polysilicon, which is completely enclosed by insulating material, usually silicon dioxide. A charge placed on the floating gate will leak off slowly enough that, for most applications, it can be considered permanent. The presence of the charge on the gate will affect the conductance of the underlying channel. In order to put a charge onto or remove charge from the floating gate some method of charge transport across the oxide is needed.

The use of an insulated-gate FET with a floating gate as a memory element was first proposed by Kahng and Sze in 1967 [13]. They built an experimental floating gate transistor that used Fowler-Nordheim tunneling to transport charge to and from the floating gate. Although the device worked, the thin oxide layer (≈50Å) prevented practical use of the device at that time.

FAMOS Transistor. The floating gate avalanche injection MOS (FAMOS) transistor was used in the first large capacity erasable programmable read-only memories [14]. Avalanche injection of electrons from the surface depletion region of a p-n junction is used to transport electrons to the floating gate of a FAMOS transistor. Charge removal is accomplished by exposing the transistor to ultraviolet radiation with a photon energy in excess of 4.3 eV. The photo current induced by the radiation will allow the charge on the floating gate to flow to the substrate.

Electrically Erasable Floating Gate Transistors. A floating gate transistor may be programmed by avalanche injection (as is the FAMOS transistor) or hot electron injection (where the source of the carriers is the channel current) and erased using Fowler-Nordheim tunneling. Mann [15] investigated three different types of floating gate transistors in both N- and P-channel versions. His transistors were all fabricated in the MOSIS low-noise analog CMOS process. His results indicate that Fowler-Nordheim tunneling current becomes detectable at 12 Volts potential difference between the poly layers. Mann attributes the low tunneling voltage to surface irregularities on the upper surface of the poly1 layer.

Floating gate transistors can be both programmed and erased by Fowler-Nordheim tunneling. Dense (4-Mbit) memory

circuits and 4-state memory cells that are both programmed and erased by Fowler-Nordheim tunneling have been fabricated [16,17].

Trimming analog circuits is another application of floating gate transistors that use Fowler-Nordheim tunneling for both programming and erasure. Thomsen and Brooke [18] fabricated such a transistor in a MOSIS 2-μm, P-well, double-poly process. The transistor was characterized by a 100-fF coupling capacitor and an injector fabricated with orthogonal strips of 2-μm poly2 (the injector input) and poly1 (attached to the floating gate). It was experimentally determined that current would flow in either direction when a high voltage of greater than 12 Volts was applied between the two polysilicon layers. Thomsen and Brooke attribute the low tunneling voltage to oxide thinning at the edges of the lower polysilicon structure and field enhancement due to the sharp edges of the lower polysilicon structure. In the course of repeatedly programming a cell, the threshold swing for constant programming voltage was reduced due to trapped electrons in the tunneling oxide after 1,000 cycles. This effect was significantly reduced by increasing the rise-time of the programming pulse to 1 mS.

Summary.

Supporting weapons systems with replacement integrated circuit types which are obsolete and no longer in production

is a significant problem. There are several methods of ensuring that needed components are available. These methods include lifetime buys, technology transfer, substitution of compatible device types, reverse engineering and remanufacturing, emulation with device arrays, and emulation with programmable components.

Fowler-Nordheim tunneling allows a current to flow across an insulating layer of silicon dioxide when the applied electric field is high enough to overcome the barrier energy. This effect may be used to transfer charge to a floating gate which controls the current flow in a field-effect transistor. The transport of charge to or from the floating gate can be used to program or erase the floating gate transistor.

III. First Demonstration Microcircuit

Microcircuit 1 Design

Floating Gate Cell Design. The cell design chosen is similar to the cell described by Thomsen and Brooke [18]. This design, shown in Figure 1, is characterized by the use of a rectangular coupling capacitor made by overlapping the second polysilicon layer (poly2) over the first polysilicon layer (poly1). Dual tunneling injectors are used to demonstrate a method of eliminating the requirement that on-die programming circuits handle high positive and negative voltages at the same nodes. The gate of the N-channel FET is connected to the lower plate of the capacitor and both injector structures. The capacitor area (area of the poly1 capacitor plate) used for all logic circuits is 144 square-microns.

The tunneling injectors are orthogonal 2-micron strips of poly1 and poly2. Fowler-Nordheim tunneling occurs due to field enhancement and oxide thinning at the edges of the poly1 strip. This allows the addition of electrons to the floating gate structure (erasing), which will raise the effective threshold voltage of the transistor, or the removal of electrons from the structure, which will lower the effective threshold voltage to a value of less than zero volts (programming).

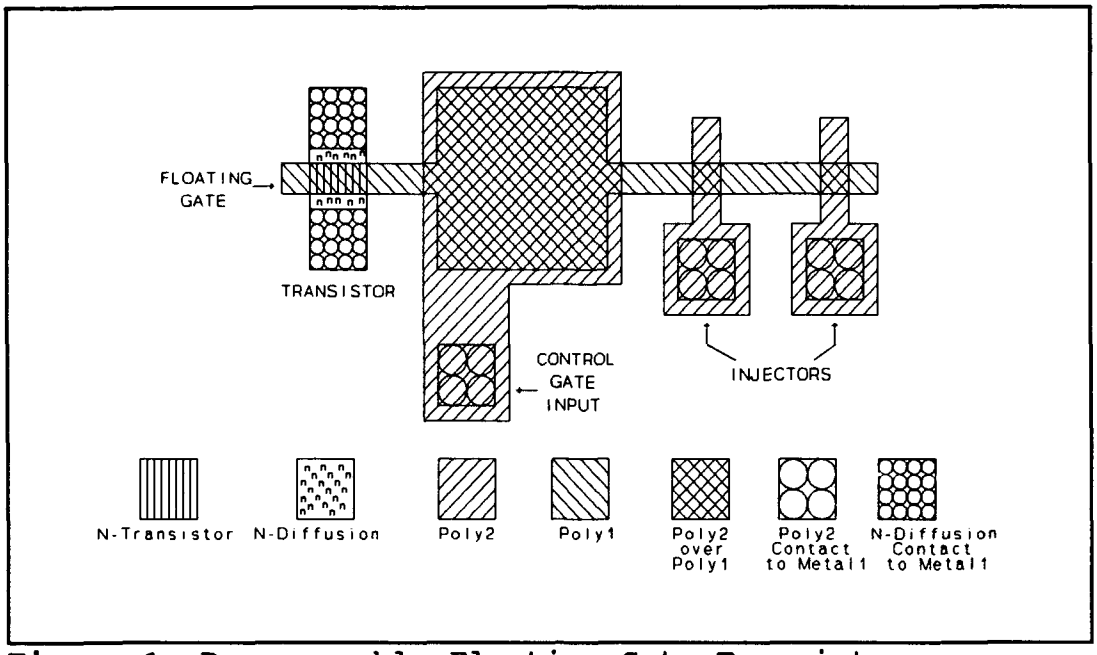


Figure 1. Programmable Floating Gate Transistor.

During normal operation, the poly2 gate will be held at ground potential. A transistor with a positively charged floating gate will be in the conducting state at this time because the threshold voltage will be less than zero volts. A transistor with an uncharged, or negatively charged, floating gate will be in a non-conducting state.

Test Cells. An array of ten programmable floating gate transistors is included to enable the direct measurement of the programming characteristics of the devices. In the array there are five different sizes of coupling capacitors represented with poly1 areas of 16, 64, 144 (same as the logic cells on the die), 256, and 576 square microns. Each of the cells share control gate and injector connections. Each drain is connected to a separate pin on the package to

allow the drain current to be measured while varying the control gate voltage.

Programmable Logic Top Level Design. The programmable logic consists of two programmable 6-input NOR gates (input level) with true and complemented outputs feeding a 5-input programmable NOR gate (output level). Figure 2 shows the logical equivalent of the programmable logic circuit where each programmable cell is equivalent to an open switch when erased and a closed switch when programmed. When the inverted outputs of the first level NOR gates are selected, a sum-of-products output is generated for any input function with two or fewer products. The intermediate output OCB0 is provided to allow direct monitoring of the wired-AND node of one of the two input gates.

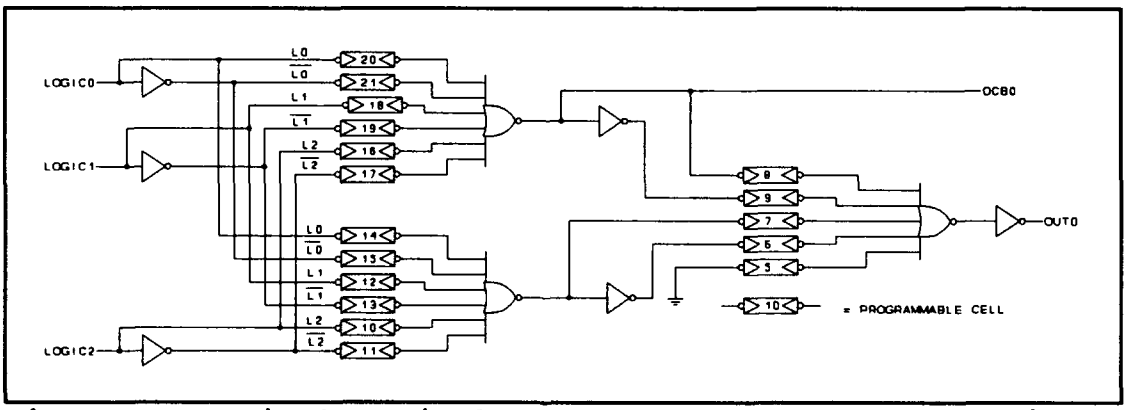


Figure 2. Logical Equivalent of the Programmable Logic of Microcircuit 1.

Shift Register. Programmable cells are selected for programming by loading a twenty-two bit word into a shift

register. Each bit is connected to the control gate of a floating gate cell. The state of this bit will determine whether a positive charge will be placed on the floating gate when a positive programming voltage is applied to an injector. If the bit is a high then programming will not take place due to the lower floating gate to programming injector potential difference.

The shift register is made up of a series of delay elements. Each delay element, shown schematically in Figure 3, is a static one-bit latch that is clocked by a two-phase non-overlapping clock. The design consists of a transmission gate multiplexer that selects either the serial data input (when SHIFT is true) or an input intended to allow verifying the status of the programmed bit in programmable inverter cells. The output of the multiplexer is gated through transmission gate X1 to node CNODE1 when the phase-1 clock is true. When the phase-1 clock becomes false, inverter X2 and clocked inverter X5 maintain the state of node CNODE1 indefinitely. The logic state of CNODE1 is inverted by X2 and is passed on to node CNODE2 by transmission gate X3 when the phase-2 clock is true. When the phase-2 clock is false, CNODE2 is maintained by inverter X4 and clocked inverter X6. The output of the delay element is the inverted state of CNODE2 appearing at the output of inverter X4.

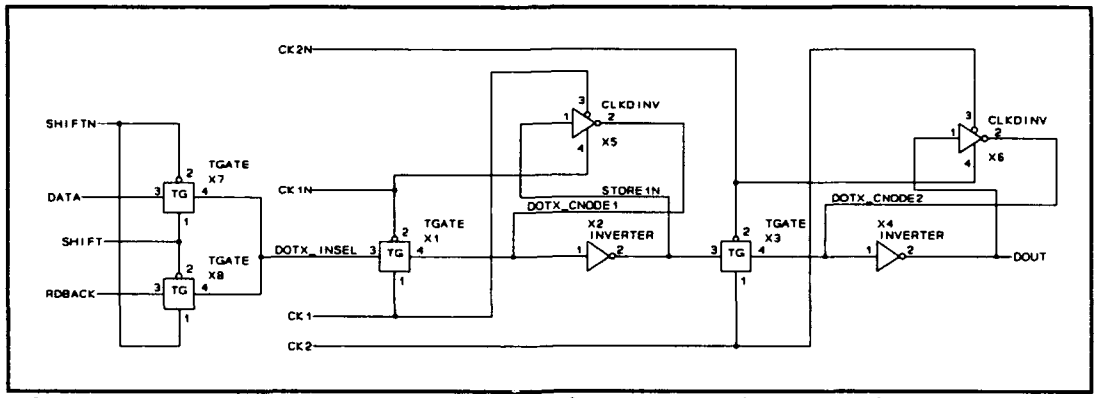


Figure 3. Delay Element Used in the Shift Register.

Programmable Cells. This microcircuit contains two different types of programmable cells. The first type of cell is shown schematically in Figure 4. The output of the delay element is connected to the poly2 control gate of the floating gate programmable transistor X4. The drain of X4 is connected to the source of N-channel transistor X3. The gate of X3 is the logic input to the cell and the drain of X3 is the logic output. If X4 is erased (in the off state when the control gate is at ground potential) then the logic input TERMIN will have no effect and the drain of X3 will always be a high impedance node. If X4 is programmed (in the on state when the control gate is at ground potential) then the drain of X3 will present a low impedance when TERMIN is high, and a high impedance otherwise.

The other type of cell (used only in a special test circuit) consists of a programmable inverter connected both to the gate of an N-channel transistor and to the RDBACK input of the delay element. This cell type is included to

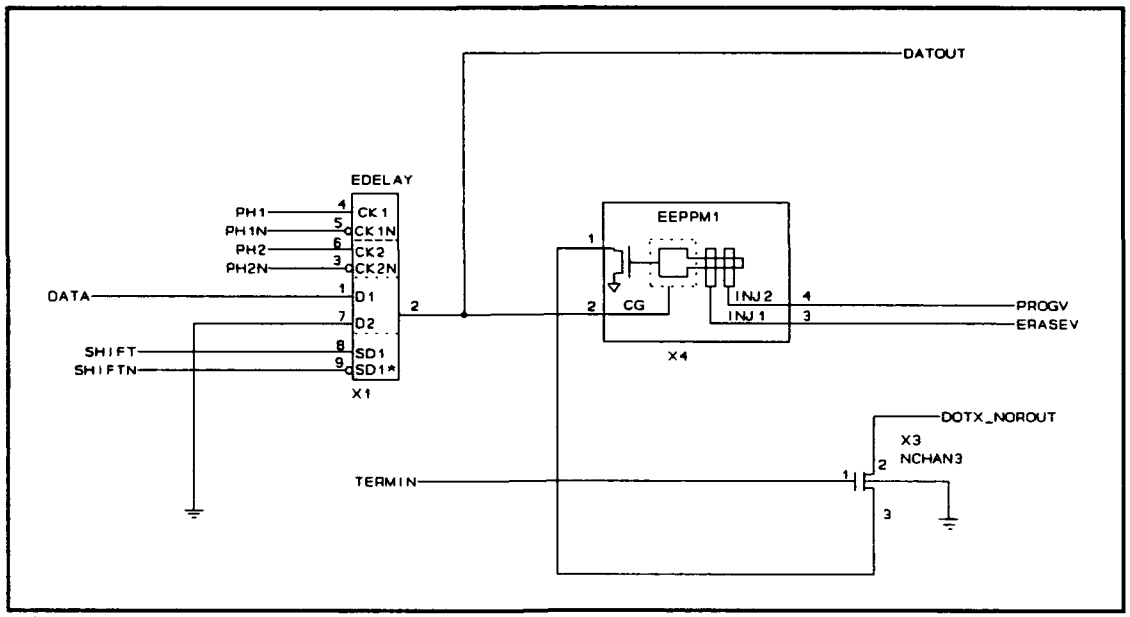


Figure 4. Programmable Cell.

allow the testing of a cell that can be read back to verify the data programmed. This cell is shown schematically in Appendix A, Figure 17.

Microcircuit 1 Detailed Design. The complete schematic of the microcircuit is included in Appendix A, Figure 11 through Figure 15. Schematic capture was performed using the Omation (Richardson, TX) SCHEMA III schematic processing program which runs on an MS-DOS computer. This package automatically generates net list information in several formats and performs some design rule checks.

Microcircuit 1 VHDL Model. The VHDL model developed for Microcircuit 1 consists of fifteen modules and is included as Appendix B. The Zycad-developed MVL7 signal type is used exclusively in all modules. This type allows for seven

signal levels, '1', 'H' (a weak '1'), 'Z' (open-circuit), 'L' (a weak '0'), '0', 'X' (a strong unknown), and 'W' (a weak unknown). These modules form a hierarchy of design units (Table 1) that parallels the hierarchy of the schematic diagrams of the design. The highest level module is the test bench (CH1BNCH) which is a mixed structural and behavioral model. The next level consists of a single module, CHIP1, which is a pure structural VHDL module. The third level down consists of structural modules ECELL1 and ECELL2 which are both pure structural descriptions. The fourth level consists of a single pure structural module, EDELAY, that is used by both ECELL1 and ECELL2.

The lowest level behavioral models vary in complexity. Three of the models (CLKDINV, INJCTR, and TPAD) perform no function but are required to allow the complete design to be analyzed and simulated. The programmable device models (EEPPM1 and EEINV) use standard logic signals as inputs to the injectors and are mechanized as latches that are programmed or erased based on the states of the injector and control gate inputs. The transmission gate (TGATE) is modeled as a one-way device for simplicity. There are two models of an N-channel transistor, NCHAN and NCHAN3. The module NCHAN models a transistor used as an open-drain inverter, while the module NCHAN3 models a transistor used as a switch. The only model of a P-channel transistor,

Table 1. Hierarchy of VHDL Models for Microcircuit 1.

Model Name	Level	Model Type	Drawing
CH1BNCH	1	Mixed	None
CHIP1	2	Structural	CHIP1
ECELL1	3	Structural	ECELL1
ECELL2	3	Structural	ECELL2
EDELAY	4	Structural	EDELAY
EEPPM1	5	Behavioral	None
EEINV	5	Behavioral	None
CLKDINV	5	Behavioral	None
INJ	5	Behavioral	None
INVERTER	5	Behavioral	None
NCHAN	5	Behavioral	None
NCHAN3	5	Behavioral	None
TGATE	5	Behavioral	None
PCHPU	5	Behavioral	None
TPAD	5	Behavioral	None

PCHPU, models a transistor used as a pull-up resistor. The remaining lowest level model, INVERTER, performs the inversion function.

The scope of the testing performed by the test bench is limited to erasing, programming, and exercising the programmable logic function. Functions exercised are three input NAND, NOR, AND, and OR functions as well as two input XOR

and XNOR functions. In addition, AND and NOR functions using the intermediate output OCB0 are included.

Each pure structural model was generated by the Automatic VHDL Generator (AVG) program (Appendix C) that the author developed using the Omation SCHEMA SPICE net list generator as a prototype. This program is written in the Microsoft QBASIC language (included as a component of the standard Microsoft DOS 5.0 package). The AVG program accepts a SCHEMA pin list format file generated by SCHPOST, a utility included in the SCHEMA package, as an input and generates both a complete VHDL structural description of the design unit and a component description of the design unit to be used by higher level units. The program also requires a component description of each type of component used in the module to be present on the current directory. Editing of the CHIP1 module VHDL module was necessary to remove references to the component P1 (the IC package) to prevent errors during the VHDL analysis phase.

Test Apparatus

Figure 5 shows the setup used to test Microcircuit 1. The test fixture (Appendix D, Figure 19 and Figure 20) is a device that provides the interface to the various functions of Microcircuit 1.

Programming Voltage Generation. Both positive and negative variable voltage programming pulses are required to deter-

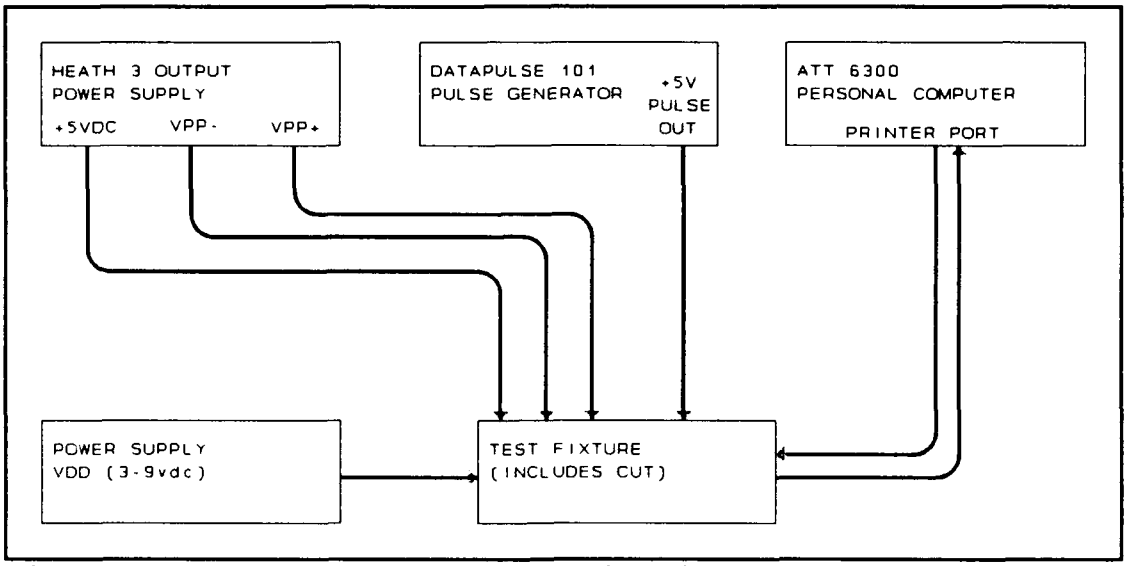


Figure 5. Test Setup for Microcircuit 1.

mine the programming characteristics of the programmable cells and to program the programmable logic. Pulse timing is provided by a Datapulse 101 pulse generator that has a single-pulse feature that provides one pulse of a preset duration each time a button is pressed. Voltage translation is provided by circuits provided on the test fixture and shown in Figure 6. Positive programming pulses are provided by one section of X7, an SN7407 non-inverting open-collector high voltage buffer. The output of this buffer is pulled up to the variable voltage VPP+ power source by R24. Negative programming pulses are generated by Q1, a 2N3906 PNP transistor, which is connected as a common-emitter inverter. Diodes CR1, CR2, and CR3 ensure that the base-emitter junction of Q1 is biased sufficiently negative to drive Q1 into saturation when the output at pin 12 of X7 is at or near

zero volts. When pin 12 of X7 is high (for example, when a positive pulse is present at the input) the base of Q1 will be driven positive, reverse biasing the base-emitter junction, allowing the collector voltage to become equal to the input voltage VPP-.

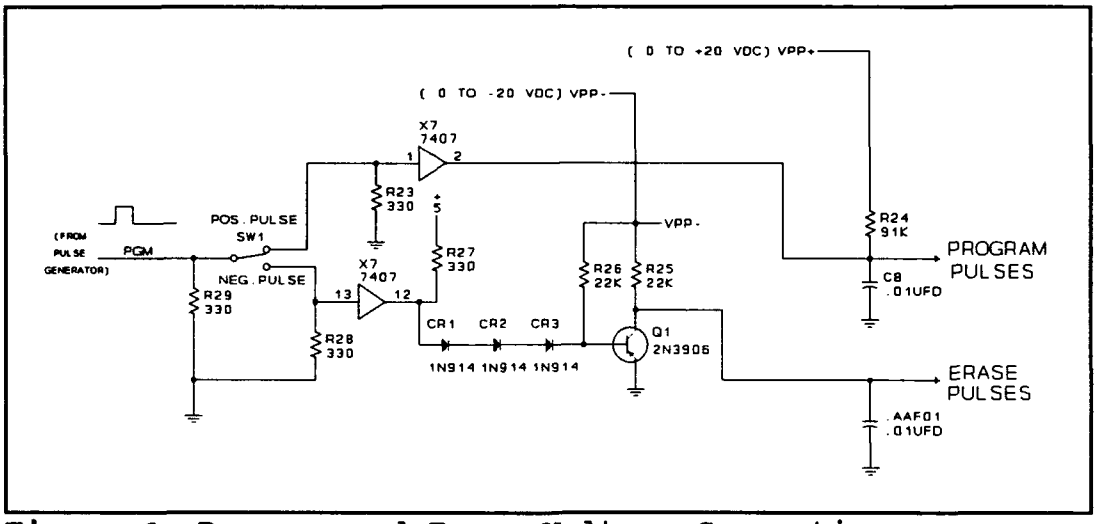


Figure 6. Program and Erase Voltage Generation.

Test Cell Interface. The interface to the test cell array consists of an input, ten outputs, and the two injectors. The control gate input is connected to a bias supply that can be varied from -8 Volts to + 8 Volts. Each output is pulled up to VDD by a 91K Ohm resistor and is connected to the input of a 74C04 inverter. The output of each inverter is inverted again and then drives a Light Emitting Diode (LED). Thus, when the control voltage input to a given cell exceeds the turn on threshold sufficiently to pull down the output, the associated LED will illuminate.

Shift Register Interface. The test fixture accepts inputs via a standard parallel printer connector from the printer port on an IBM PC or compatible computer. The main input signals associated with the shift register (SHIFT, PH1, PH2, and SDATIN) are buffered from the computer interface. Derived signals SHIFTN, PH1N, and PH2N are inverted signals SHIFT, PH1 and PH2 respectively. Each signal is driven by an open-collector driver that is pulled up to VDD. This arrangement allows the signals to transition between ground and VDD even when VDD varies significantly from +5 Volts.

Programmable Logic Interface. The three logic inputs, LOGIC0, LOGIC1, and LOGIC2, are buffered from the computer interface identically to the method used for the main shift register signals. One output signal, OUT0, is inverted and fed back to an input on the computer interface. The other signal, OCB0, is a connection to a high-impedance internal signal and is buffered by a CMOS inverter, the output of which is buffered and fed to the computer interface.

Interface Software. A computer program that provides a convenient menu driven user interface to control the test fixture was written in Borland Turbo Pascal and is included as Appendix E. Functions supported include individual signal control as well as composite operations such as loading the shift register with a program word, exercising

the programmable logic function, and displaying the resultant truth table.

Test Results

VHDL Results. The VHDL model was programmed with eight different program words. Six program words were selected to exercise the complete programmable function. The remaining two program words were selected to test the intermediate node OCB0. The program words used and the actual results are shown in Table 2 for output OUT0 (the main programmable logic output) and Table 3 for the intermediate output, OCB0. In both cases the actual results are identical with the expected results.

Table 2. VHDL Simulation Results for Output OUT0

Inputs			Outputs (OUT0)					
Program Word			00A840H	05480H	00A880H	05440H	246180H	285180H
L2	L1	L0	NAND	NOR	AND	OR	XOR(0,1)	XNOR(0,1)
0	0	0	1	1	0	0	0	1
0	0	1	1	0	0	1	1	0
0	1	0	1	0	0	1	1	0
0	1	1	1	0	0	1	0	1
1	0	0	1	0	0	1	0	1
1	0	1	1	0	0	1	1	0
1	1	0	1	0	0	1	1	0
1	1	1	0	0	1	1	0	1

Table 3. VHDL Simulation Results for Output OCB0

Inputs			Outputs (OCB0)	
Program Word			2A0000H	150000H
L2	L1	L0	AND	NOR
0	0	0	0	1
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	0

Microcircuit Test. Microcircuit 1 was fabricated in the Orbit Semiconductor low noise analog 2-micron CMOS N-well process through MOSIS. Six circuits were chosen for test cases and marked as numbers one through six. Initial testing indicated that circuit #2 could not be erased or programmed and circuit #5 exhibited leakage on an injector input. Both of these circuits were excluded from further tests. During the initial testing, it was found that TCELL1 output (pin 4) was not connected internally due to a layout error. Cell TCELL6 was not connected to the INJCTR1 or INJCTR2 lines in the circuit due to another layout error. TCELL1 and TCELL6 were therefore excluded from testing.

In order to support the testing of Microcircuit 2, the test cell characterization concentrated on determining a practical set of programming and erasing methods for a cell that has +5 Volts applied to the control gate during normal operation. The four remaining test circuits were tested with several different VPP+, VPP-, and gate voltages. The test data is included in Appendix K. Chips #3, #4, and #6 were programmed at 12.75 Volts VPP+ with 0V control gate potential. Table 4 contains some statistical measures of the programming results of cells exclusive of TCELL9 and TCELL10, the cells with the smallest capacitor size. The mean threshold voltage, variance, standard deviation, maximum, and minimum values are tabulated for eighteen cells.

One unexpected result was noted during the testing on Chip #4. The threshold voltage of TCELL9, a minimum coupling cell with a capacitor area of sixteen square-microns, had been lowered to less than -6.0 V. and then apparently increased without any programming pulses being applied. TCELL9 on Chip #4 had the largest threshold shift of any of the test cells on Chip #4. It was noted that the threshold changed when the control gate was lowered to -8.1 Volts. After ten seconds, the threshold of TCELL9 had increased to -3.4 Volts. No change in threshold voltage was noted when the control gate input was held at zero volts. A possible explanation is that Fowler-Nordheim tunneling from the

Table 4. Programming Result Statistics for Samples #3, #4, and #6, with +12.75 Volts VPP+ and 0 Volts Control Gate Potential.

T (mS)	Mean (V)	Variance (V)	Std.Dev. (V)	Max (V)	Min (V)
1	3.72	1.79	1.34	6.6	1.8
2	3.20	1.98	1.41	6.15	1.1
5	2.71	2.01	1.42	5.7	0.6
10	2.33	2.01	1.42	5.2	0.2
20	2.02	2.02	1.42	5	0.0
50	1.60	2.08	1.44	4.5	-0.5
100	1.35	2.07	1.44	4.3	-0.8
200	1.07	2.01	1.42	4.1	-1.0
500	0.74	1.97	1.41	3.7	-1.3
1000	0.49	1.77	1.33	3.4	-1.4

negative control gate to the positive floating gate was occurring, causing the threshold voltage of the cell to change.

Programmable Logic Test. During initial testing it was determined that VDD was not connected to X33 or X34 due to a layout error. This made the main logic output, OUT0, unreadable. Intermediate node OCB0 was still available and the logic function was tested up to that point. Programming was attempted at 12.75 V VPP+ and did not work. VPP+ was increased on each chip tested until the chip programmed or reached a state where a zero output appeared where a one should be in the truth table indicating at least one input

which should not be programmed was programmed. The results of the logic test are summarized in Table 5.

Table 5. Logic Test Results.

UUT	Function	VPP+	Time (mS)
#1	NOR	14.75	500
#1	AND	16.00	500
#3	NOR	14.75	500
#3	AND	N/A	N/A
#4	NOR	14.75	500
#4	AND	14.75	500
#6	NOR	14.25	500
#6	AND	14.25	200

Summary

An integrated circuit to demonstrate programmable logic functions was designed. The design used logic cells which are programmed and erased using Fowler-Nordheim tunneling. A VHDL simulation was performed on the logic portions of the circuit.

The microcircuit was fabricated in the Orbit Semiconductor 2-micron N-well low noise analog CMOS process through MOSIS. Test cells were erased and programmed using several VPP+ voltage levels. Some evidence of Fowler-Nordheim tunneling from the control gate to the floating gate was noted

in one case. Test results indicate that the programmability of the floating gate transistors used in the test cells varies widely. The variability of programming characteristics may result in a cell which will not program when selected for programming, or a cell which will program when not selected for programming. The programming profile used (VPP+ magnitude, control gate potential, and duration of the programming pulse) must ensure that the least programmable cell selected for programming is programmed, and that the most programmable cell that is not selected for programming is not programmed.

An intermediate logic output was programmed and tested. Unexpectedly high voltages were necessary to program the logic, possibly due to the variability of the programmability of the cells. The main circuits did not have a method of reading back the state of the programmed cells. Such a capability would simplify the programming task.

IV. Microcircuit 2

Architecture

Emulation Target. Microcircuit 2 is designed to emulate the logic function of simple TTL combinational logic chips. Fourteen pins of the forty-pin Microcircuit 2 package are emulation logic input-output (I/O) pins that are mapped to the logic I/O pins of the emulated TTL circuit. Two different maps are supported depending on the number of logic I/O pins on the target circuit. The first mapping, which applies to TTL circuits packaged in a sixteen pin dual-in-line-package (DIP) (fourteen logic I/O pins, VCC and Ground), maps all fourteen emulation logic I/O pins to the fourteen logic I/O pins of the target circuit. The second mapping, for TTL circuits packaged in a fourteen pin DIP, maps twelve of the fourteen emulation logic I/O pins to the twelve logic I/O pins of the target circuit. Pin mapping is shown in Table 6. A representative group of the TTL types that Microcircuit 2 can emulate and for which logic emulation was demonstrated is listed in Table 7.

Emulation Limitations. The emulation performed is of the logic function only. Other parameters such as drive capability, input source current, output drive capability, and delay time are not emulated.

Emulation I/O Pins. Each emulation I/O pin can be programmed as an input only (output disabled), as an open drain

Table 6. Pin Mapping of TTL I/O Pins to Microcircuit 2 Emulation Pins.

	TTL Pkg. Type		Pin #		
Name	16 Pin	14 Pin	Actual	Group	Section
PINAL	1	1	18	Left	Top
PINBL	2	2	19	Left	Top
PINCL	3	3	20	Left	Top
PINDL	4	4	21	Left	Both
PINEL	5	5	22	Left	Bottom
PINFL	6	6	23	Left	Bottom
PINGL	7	None	24	Left	Bottom
PINAR	15	13	2	Right	Top
PINBR	14	12	1	Right	Top
PINCR	13	11	40	Right	Top
PINDR	12	10	39	Right	Both
PINER	11	9	38	Right	Bottom
PINFR	10	8	37	Right	Bottom
PINGR	9	None	36	Right	Bottom

output to emulate an open-collector output, or as a CMOS totem-pole output to emulate a TTL totem-pole output. Tri-state outputs are not emulated by Microcircuit 2.

Programmable Logic. The programmable logic of Microcircuit 2 is arranged as an OR-OR array with individually invertible outputs. Figure 7 presents a high-level logical equivalent circuit of Microcircuit 2. The input logic is divided into two groups, Left and Right, referring to the

Table 7. TTL Chip Emulations to be demonstrated.

TTL Type	Variants	# Pins	Function
00	L,H,STD.	14	Quad 2-Input NAND
03	L,H,STD	14	Quad 2-Input O.C. NAND
04	L,H,STD	14	Hex Inverter
08	L,H,STD	14	Quad 2-Input AND
20	L,H,STD	14	Dual 4-Input NAND
27	L,H,STD	14	Triple 3-Input NOR
133	Schottky	16	13-Input NAND

emulated pin positions on a TTL Dual-Inline-Package (DIP). The two input groups are further divided into Top and Bottom sections, again referring to the position of the emulated pins on a TTL DIP. Each section consists of two programmable eight-input OR gates with true and complemented inputs from a group of four pins. One pin in each group belongs to both sections of the group.

An OR-OR array is a degenerate form of two-level logic. The non-unique output state of the first level (high) is the same as the controlling input value of the second level. Any high input to the second level will cause the output to become high. Therefore the OR-OR array with invertible inputs and outputs can perform the AND, OR, NAND, or NOR of any combination of the input signals or complements. This type of array cannot perform an exclusive-OR (XOR) or AND-

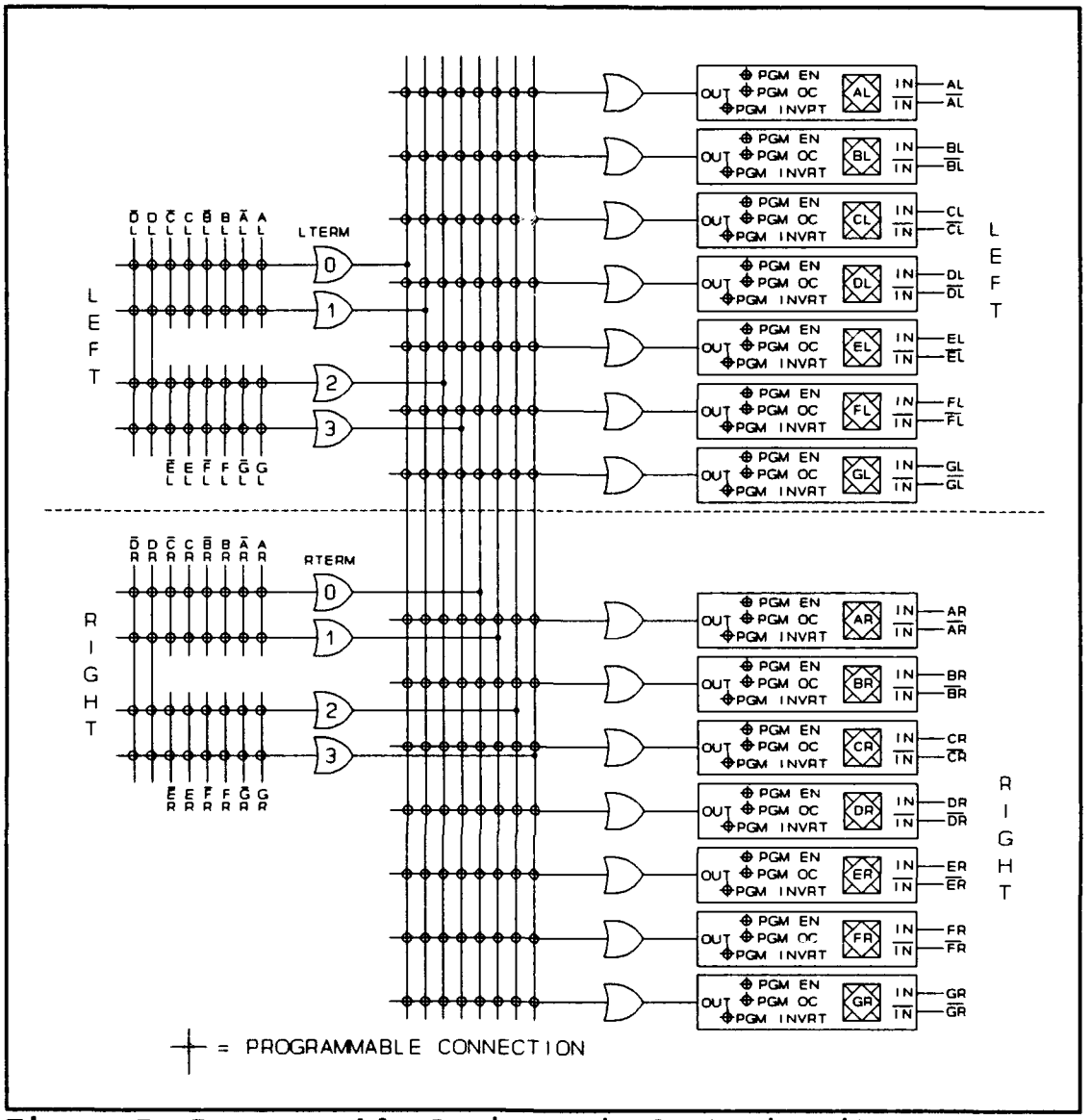


Figure 7. Programmable Logic Equivalent Circuit.

OR-INVERT of the inputs.

Programmable Cell. Each programmable cell consists of a two input NAND gate with the output connected to the control gate of a programmable floating gate transistor (Figure 8). Each cell is initially erased by applying a negative voltage high enough to cause tunneling to the floating gate for a

sufficient duration to raise the threshold voltage above +5 Volts. One or both inputs to the NAND gate should be low, causing the control gate to be high, to reduce the magnitude of the negative potential required for erasure.

Programming the transistor to be in the conducting state when the gate voltage is at VDD requires that the control gate be held at ground potential while an injector is raised to a high positive voltage. The NAND gate provides a method of addressing each cell by the correspondence of two signals. If the output of the NAND gate is near ground then the voltage induced on the floating gate by the control gate will be near zero, causing the highest possible potential difference between the floating gate and the positive injector input. If the cell is not being addressed for programming, then one or both NAND gate inputs will be low and the output of the NAND gate and the control gate of the floating gate transistor will be at VDD potential.

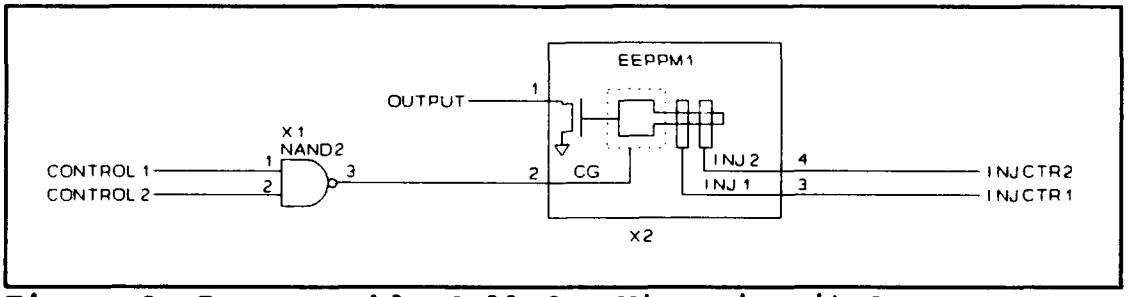


Figure 8. Programmable Cell for Microcircuit 2.

Programming. Microcircuit 2 functions in one of two modes, normal operating mode and programming mode. The mode is selected by the OPERATE input to the chip. When OPERATE is false all emulation outputs are disabled and three of the emulation pins, PINBL, PINCL, and PINAL, become the inverted serial data, phase-1 clock, and phase-2 clock inputs respectively. Programmable cells are programmed in groups of eight by loading a thirty-six bit serial Program Word into the shift register. The first twenty-eight bits loaded (Select Word) should be all zeros except a unique select bit that determines which group of eight cells will be programmed. The last eight bits loaded make up the Global Program Word (GPW). The GPW individually enables cells in the group of eight cells selected by the Select Word to be programmed. Figure 9 shows the logical equivalent of the shift register. The programming injectors are then pulsed at the VPP+ level, programming the selected cells. Erasing all cells simultaneously is accomplished by loading all zeros into the shift register and pulsing the erase injectors to VPP-.

Design of Microcircuit 2

Presentation. The design of Microcircuit 2 consists of a set of modules each of which describes a component. The design hierarchy will be presented, then each design unit will be discussed proceeding from the lowest hierarchical level to the highest. The VHDL model for each design unit

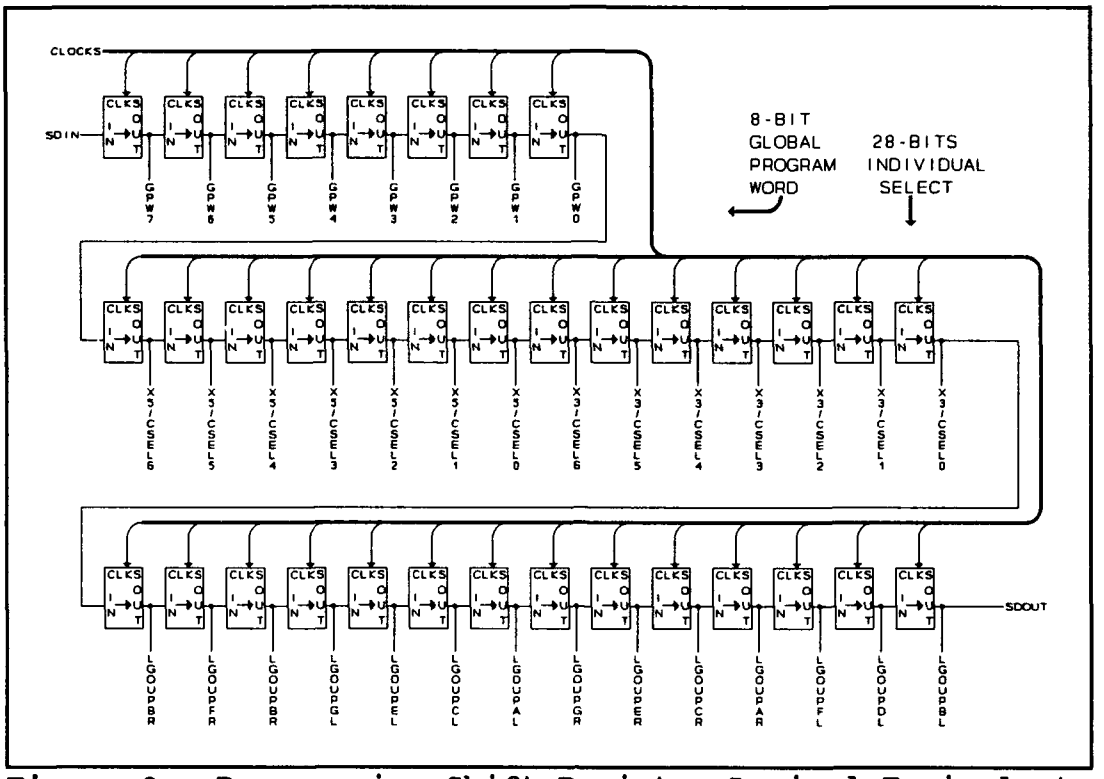


Figure 9. Programming Shift Register Logical Equivalent.

that has an associated drawing was automatically generated by the AVG program previously described.

Design Hierarchy. Table 8 shows the hierarchy of design drawings and VHDL models. This division of the design evolved with the actual physical design of the integrated circuit such that design drawings, VHDL models, and blocks defined in the MAGIC integrated circuit layout editor correspond.

Behavioral Components. The NAND2, INVERTER, CLKDINV, TGATE, NCHAN, NCHAN3, PCHPU, and EEPPM1 components (and corresponding VHDL models) have been previously described as part of Microcircuit 1. The only added behavioral model of a compo-

Table 8. Hierarchy of Design Units for Microcircuit 2.

#	VHDL Model	Uses Model #(s)	Drawing Name	Location
1	CH2BNCH	2	None	N/A
2	ELPROJ	3-5,9,23,25	ELPROJ	Figure 21
3	ELPRO7	7,8,10,12	ELPRO7	Figure 23
4	ELSHFT14	10	ELSHFT14	Figure 26
5	ELOUT14	6	ELOUT14	Figure 25
6	ELOUT7	11	ELOUT7	Figure 27
7	ELIN2	11	ELIN2	Figure 29
8	ELFIX7	13	ELFIX7	Figure 30
9	ELSHIFT8	15	ELSHIFT8	Figure 31
10	ELSHIFT7	15	ELSHIFT7	Figure 32
11	ELOG8	14,17,23	ELOG8	Figure 33
12	ELPIN	20-25	ELPIN	Figure 34
13	ELFIX1	16,17,23,25	ELFIX1	Figure 35
14	ELOG1	16,19,25	ELOG1	Figure 36
15	ELDELAY	21,22,23	ELDELAY	Figure 37
16	EEPPM1	N/A	None	
17	PCHPU	N/A	None	
18	NCHAN	N/A	None	
19	NCHAN3	N/A	None	
20	TRIPAD	N/A	None	
21	TGATE	N/A	None	
22	CLKDINV	N/A	None	
23	INVERTER	N/A	None	
24	NOR2	N/A	None	
25	NAND2	N/A	None	

nent is TRIPAD, which consists of a physical pin and P-channel and N-channel driver transistors.

Component ELOG1. ELOG1 (Appendix F, Figure 36) is a single element of programmable logic which may, with the addition of a pull-up resistor and an inverter, be connected in parallel to build multiple input programmable OR gates. The element is programmed by making the CONTROL1 and CONTROL2 inputs high (forcing the control gate input of floating gate transistor X2 low) and applying a positive programming voltage to an injector. Erasing is accomplished by making either CONTROL1, CONTROL2, or both CONTROL1 and CONTROL2, low (causing the control gate input of floating gate transistor X2 to become high) and applying a negative erasing voltage to an injector. The output of ELOG1 is an open-drain circuit.

Component ELFIX1. The ELFIX1 circuit (Appendix F, Figure 35) is identical with ELOG1 with the addition of PCHPU X4, a P-channel transistor used as a pull-up resistor, and X3, an inverter. The output of ELFIX1 is either zero (erased) or high (programmed). The programming method for ELFIX1 is identical with the method used for ELOG1.

Component ELPIN. ELPIN (Appendix F, Figure 34) is a single emulation logic I/O pin and associated output driver and input circuits. The input is first buffered by inverter X2, which supplies the inverted input. The inverted input is again inverted to supply the non-inverted input. Five signals control the output of ELPIN. PROGEN is an overrid-

ing output disable signal intended for use when the microcircuit is being loaded or programmed. OUTEN is an output enable that is an input from a programmable cell that is programmed whenever the pin is selected for use as an output. The DPINOUT signal is the logic output signal which controls the pin when the pin is enabled as an output. INVOUT, when true, will invert the output, making it the complement of DPINOUT. OPNCLCTR, the last signal that controls the output, disables the P-channel output driver transistor when high. OPNCLCTR is an input from a programmable cell that will cause the pin to emulate a TTL open-collector output when programmed.

Simple Composite Components. Certain design units are simple composite components that consist of identical lower level structural components and, in some cases, behavioral components. The following components are simple composite components. ELOG8 (Appendix F, Figure 33) is eight ELOG1 components with a PCHPU pull-up and an inverted output. ELSHIFT7 (Appendix F, Figure 32) is seven ELDELAY components connected as a 7-bit shift register. ELSHIFT8 (Appendix F, Figure 31) is eight ELDELAY components connected as an 8-bit shift register. ELFIX7 (Appendix F, Figure 30) is seven ELFIX1 components and common connections. ELIN2 (Appendix F, Figure 29) is two ELOG8 components with parallel inputs. ELOUT7 (Appendix F, Figure 27 and Figure 28) is seven ELOG8

components with parallel inputs. ELOUT14 (Appendix F, Figure 25) is two ELOUT7 components with parallel inputs. ELSHFT14 (Appendix F, Figure 26) is two ELSHIFT7 components connected as a fourteen-bit shift register.

Complex Composite Components. Components which incorporate two or more different types of lower level structural components are complex composite components. There are two complex composite components, ELPRO7 and ELPROJ. Component ELPRO7 consists of seven ELPIN components connected to two ELIN2 components which implements one group of emulation I/O pins. Three ELFIX7 components provide the output enable, invert output, and open-drain signals to each I/O pin in the group.

The highest level component, ELPROJ, is the complete microcircuit. ELPROJ consists of two ELPRO7 components that provide the outputs and inputs for both groups of emulation I/O pins. One ELOUT14 component provides the second level of the OR-OR programmable logic for all fourteen emulation logic I/O pins. An ELSHIFT14 component provides the select bits to the ELOUT14 component and an ELSHIFT7 component provides the Global Program Word to all programmable composite components. ELPROJ also includes a logic block that controls the PH1, PH1N, PH2, and PH2N clock signals and the serial data input SDIN. As shown in Figure 10, input signal OPERATE forces PH1 and PH2 clocks true while forcing SDIN to

a low level. With both clocks true, the zero on SDIN propagates through the entire shift register causing the Program Word to be zero. The inputs on PINAR, PINBR, and PINCR (signals INLA, INLB, and INLC respectively) will have no effect on the clocks or serial data input when OPERATE is true. When input signal OPERATE is false, signal PROGEN is true and signal INLC (controlled by input PINCL) controls clock PH1; signal INLA (controlled by input PINAL) controls clock PH2; and signal INLB (controlled by input PINBR), when inverted, controls serial data input SDIN.

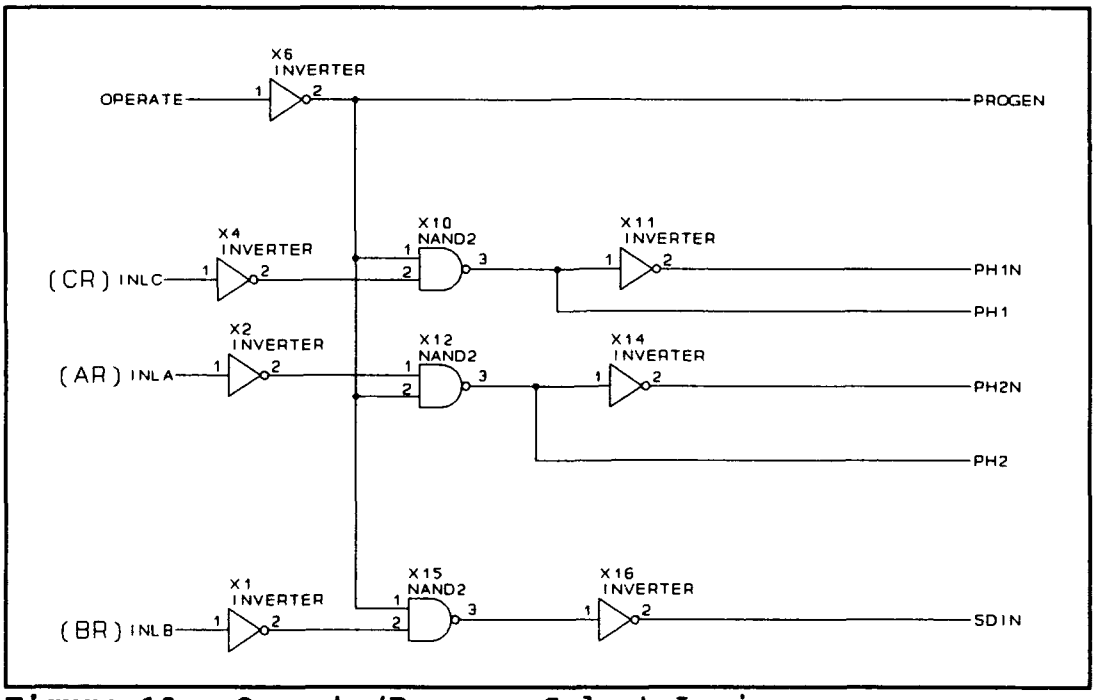


Figure 10. Operate/Program Select Logic.

Detailed Programming Model. Figures 38 through 41 in Appendix H graphically present the detailed programming model of Microcircuit 2. Each programmable connection is labeled with

the select bit number and the bit number of the GPW that selects it for programming.

Microcircuit 2 Test

Emulation Program Words. For each of the emulation targets in Table 7, a set of Program Words is needed to program both the VHDL simulation of Microcircuit 2 and the actual microcircuit. Program Words for each of the emulation targets are presented as tables in Appendix I. The Lotus Symphony spreadsheet was used to generate the GPW values directly from the table entries.

VHDL Test. All VHDL models were analyzed and simulated. The highest level module, the test bench, contains the test cases. The test bench, CH2BNCH, was executed and provided results for each of the emulated circuit types. Results of the VHDL tests are presented in tabular form in Appendix J. Each table shows the test cases tested (inputs) and the outputs obtained. All expected and actual outputs are identical and are not listed separately.

Test Fixture For Microcircuit 2. Schematic diagrams of the test fixture that was designed and fabricated to test Microcircuit 2 are presented in Appendix L, Figure 49 and Figure 50. Features of the test fixture include a computer interface, programming pulse generation and voltage translation, clock and data inputs to the UUT, and input logic level setting.

The test fixture is controlled through J1, a standard thirty-six pin parallel printer connector. High-voltage programming voltage translation circuits are similar to the circuits used on the test fixture for Microcircuit 1 and previously described in Chapter 3.

Program pulse generation is performed by X1, a 74121 mono-stable multivibrator, which generates a positive 10-mS pulse whenever the signal PULSE transitions to a high level. The pulse width may be adjusted to 10-mS using variable resistor R7. The pulse is gated by two sections of X2, a 74LS08 AND gate, such that a programming pulse is applied to the VPP- (erase voltage) translation circuit when signal ENERA is high, or to the VPP+ (programming voltage) translation circuit when signal ENPRO is true. Each of the signals which trigger the pulse and gate it to a translation circuit is an input from the control interface.

When signal OPERATE is false, serial data input and phase-1 and phase-2 clock inputs are applied to the appropriate pins on the UUT by X4, a three-state buffer. When OPERATE is true, logic levels set by multiple-section switches SW2 and SW3 are applied to the logic emulation pins. Each logic level applied is routed through a 10K-Ohm resistor so that if the individual logic emulation pin is programmed as an output, it will overcome the relatively weak input. Each logic emulation pin is connected to a test

point which allows monitoring of the level by an oscilloscope or multi-meter.

Test Software For Microcircuit 2. The control program for the Microcircuit 2 test fixture, Appendix M, is written in Borland Turbo Pascal and executes on an ATT-6300 (or equivalent IBM-PC compatible) personal computer with a printer interface base port address of 378_{16}. Functions provided include individual signal set/reset for each of the input signals, erase, clock control, input program word, write program word, program single word, input program word file name, and program with parameters in file.

Test Results For Microcircuit 2. Four chips were fabricated in the Orbit Semiconductor 2-micron N-well low noise analog CMOS process through MOSIS. The chips were marked with identifying numbers one through four. Sample #1 was tested using the test fixture and control program, with the high VDD set to +7V during programming and erasure. Proper serial register functioning was verified and several unsuccessful attempts program the chip with the 7400 emulation program words were made. Initial programming was attempted with VPP+ set to 12.25V, 11V, and 13V. After each programming attempt the logic function was tested. No consistent results were obtained. After one programming attempt, the temperature of Chip #1 increased and, after removing all

voltages applied to Chip #1 then reapplying VDD, a test of the serial register function failed.

The high VDD voltage feature of the test fixture was disabled and Chip #2 was programmed with VDD set to +5V. Programming with the full set of program words for the 7400 was not successful. A reduced set of the 7400 emulation program words was used to program Chip #2, with VPP+ set to +10.9V, such that after the application of four 10-mS pulses output PINCR performed the NAND function. The high VDD was then re-enabled and Chip #2 was programmed with the full set of 7400 emulation program words. Using VPP+ of +11.5V, outputs PINCL generated the NAND function after ten 10-mS programming pulses were applied.

After programming with ten additional 10-mS +11.5V pulses outputs PINCR and PINCL responded correctly and generated the NAND function. Proper response was not obtained from outputs PINFR and PINFL. After repeated attempts to program the complete NAND function Chip #2 became non-functional while being programmed.

Attempts to program Chip #3 and Chip #4 were made with the 7400 and 7404 emulation words. Programming was attempted at 10.5, 11.0, 11.5, 12.0, 12.5, and 13.0 Volts. Although some of the cells were being programmed as evidenced by the outputs becoming active (since when all cells are erased the

outputs are in the high-impedance state) consistent results were not obtained.

Summary

Microcircuit 2 was designed to emulate the logic function of a set of TTL components. Verification of the design was performed using a VHDL model of Microcircuit 2 which successfully emulated the TTL components. The chip was fabricated and one sample of the chip was partially functional.

One reason for the lack of uniform programmability in the fabricated circuit is a design error. The logic level applied to the SDIN signal (which sets the level of both inputs of the NAND gate which sets the control gate voltage) is low when OPERATE is true. This means that the control gate input of the floating gate transistor is at +5V during logic emulation and the threshold voltage must be above that level for the floating gate transistor to be non-conducting. The SDIN signal should have been high during operation, resulting in a low on the control gate inputs of all floating gate transistors. The VHDL model for the floating gate transistor does not use the control gate input as a factor in determining the state of the transistor except during programming and therefore did not provide visibility of the error.

V. Summary and Conclusions

Summary

Two integrated circuits were designed, fabricated, and tested to demonstrate programmable logic for use as a replacement for obsolete TTL combinational logic components. Microcircuit 1 incorporated test circuits to allow experimental determination of the programming characteristics of floating gate transistors and the demonstration of rudimentary programmable logic functions. Test results on Microcircuit 1 indicate that the programmability of floating gate transistors fabricated in the MOSIS 2-μm N-well low noise analog process varies widely even between transistors on the same chip. Microcircuit 2 was designed to emulate the logic emulation function of a group of TTL combinational logic chips. Partial functionality was demonstrated but full functionality was not achieved. The most likely reasons for the lack of full functionality are a combination of a design error (+5V applied to the control gate inputs of the floating gate transistors during normal operation) and the previously noted variability of programming characteristics of the floating gate transistors. Even so, proper logic functionality was obtained from two sections of one test circuit (Chip #3). The destruction during test of two samples of Microcircuit 2 may have been due to latchup while VDD was raised to +7V for programming.

Conclusions

Designs of programmable devices for replacement of obsolete TTL circuits can be demonstrated using circuits fabricated through MOSIS. Floating gate transistors that use Fowler-Nordheim tunneling for programming and erasing can be used as programmable cells in such devices. The variability of the programming characteristics of the programmable cells makes the selection of the proper programming profile a critical task. Control of the programming characteristics of the devices would require control over the inter-poly oxide thickness and surface smoothness of the polysilicon layers at the injectors.

Recommendations.

Microcircuit 1. Further tests could be made on Microcircuit 1. Only six of the twelve circuits fabricated were tested, leaving six circuits that have never been programmed. Appropriate tests can be devised that use the test cells on the circuits to measure the effects of electron trapping in the oxide layer on programming characteristics. Since the tunneling characteristics of the devices are not uniform, measurement of programming characteristics at lower programming voltages (using long programming pulses) and higher programming voltages (using short pulses) may assist in determining a more repeatable programming profile for the devices.

Microcircuit 2. Several simple changes to the Microcircuit 2 design should allow the circuit to function properly. Changes recommended are connecting the phase-1 clock, phase-2 clock, and the serial data inputs to input pins to allow direct external control of these signals. In addition, the eight intermediate logic signals, RTERM0-3 and LTERM0-3, should be connected to output pins to enhance testability.

Another change, which would require more extensive modifications to the layout, would be to add a programmable inversion stage to each intermediate logic term. This would increase the number of TTL types that can be emulated, including allowing the emulation of the XOR and XNOR functions.

One other modification, which might increase the programmability of the circuit, would be to provide a global control gate voltage that could be varied between ground and VDD potentials. This voltage would be routed to every programmable cell, possibly replacing either the VDD or ground connection on the NAND gates that drive the control gate. The control gate voltage could then be varied, allowing different programming profiles to be developed and tested.

Complete Emulation. To emulate a TTL component completely, the replacement device should meet the specifications of the

original device and function identically in a circuit designed to operate with the TTL component. Microcircuit 2 does not match the drive, input current, or timing of the emulated circuits. Three additions to the design would be needed to emulate a group of TTL components completely.

First, an I/O circuit that can emulate the characteristics of a TTL input or output is needed. Such a design, in a Bi-CMOS technology, could use bipolar components for input sensing and output driving. Certain characteristics of the I/O pin should be programmable. Suggested programmable parameters are output drive capability, input pin source current, input hysterisis, and output delay. Practical considerations may limit initial research to technologies available through MOSIS even though the characteristics of the bipolar transistors that can be fabricated in processes presently available through MOSIS may not allow emulation of any but the slowest TTL family.

Second would be a voltage comparator that can sense the absolute supply voltage. An actual emulation circuit has only the emulated I/O pins and power and ground pins. There are no pins available that can be dedicated to selecting the mode of operation. If the circuit can sense that the voltage applied to the VCC pin exceeds some voltage (which also exceeds the +7V absolute maximum rating of TTL circuits), then the sensed condition can set the operating mode to

program/erase and the circuit I/O pins can be used to control programming and erasure. This is the same as the function of the OPERATE input to Microcircuit 2.

The last addition to the design is a programming and erasing pulse generation and control circuit. Components of this circuit include a positive high voltage supply, a negative high voltage supply, and switching circuits to control pulse gating. The high voltage generation circuits can be charge pumps that rely on externally generated clocks that are fed to the circuit through I/O pins. Pulse gating and timing can also be controlled externally. The current required from the high voltage circuits is low since programming and erasing are performed using Fowler-Nordheim tunneling.

Advanced Architecture. The architecture of Microcircuit 2, even with the addition of invertible intermediate terms, is limited in the number of part types it can emulate. An advanced architecture should allow the emulation of a much larger number of TTL part types. Multiplexers, decoders, sequential circuits (including flip-flops, counters, and shift registers), and special purpose circuits are candidates for emulation. So as to emulate the largest possible number of TTL components on a small die, the architecture may become more special purpose and have features that support one or several special cases. Alternately, several

different architectures, each of which supports a certain class of components, may be a more effective solution.

Schematic to VHDL. The program developed to generate VHDL structural descriptions from a pin list generated by the SCHEMA post-processing utility (Appendix C) proved to be very useful by greatly reducing the time needed to generate the models for simulation. The program is limited to signal types contained in MVL7 and resolved type DOTX and does little error checking. Either developing this program or identifying and obtaining a commercial program that performs the same function would be useful.

Automated Design Checking. The design of Microcircuit 2 evolved such that there was a general correspondence of design modules (both VHDL models and drawings) and MAGIC layout blocks. Given such a correspondence, it should be possible to extract the design from the MAGIC layout and automatically compare the layout net list to the design net list. Such checking should greatly decrease the likelihood of layout errors and the development of whatever software is needed to make the net list comparisons is recommended.

Lessons Learned

Testability. The design of Microcircuit 2 included the logic for pin sharing during programming and erasure. Since the shift-register level was incorrect during operation due to a design error, test results were limited. Using separate

dedicated pins for programming control would have allowed full testing of the microcircuit. Also, connecting intermediate outputs to I/O pins (perhaps even straight connections to allow externally over-riding the levels) would have enhanced testability. Since there were unused pins on the package, both of these changes could have been implemented.

Microcircuit 1 included certain circuits to allow reading back the state of a programmed cell. These circuits were not tested; however, since each cell was associated with a shift-register element, the die area taken up by such a circuit is excessive. There remains a need for an area efficient method to determine the state of each programmable cell in the circuit.

Appendix A: Microcircuit 1

Schematic Diagrams.

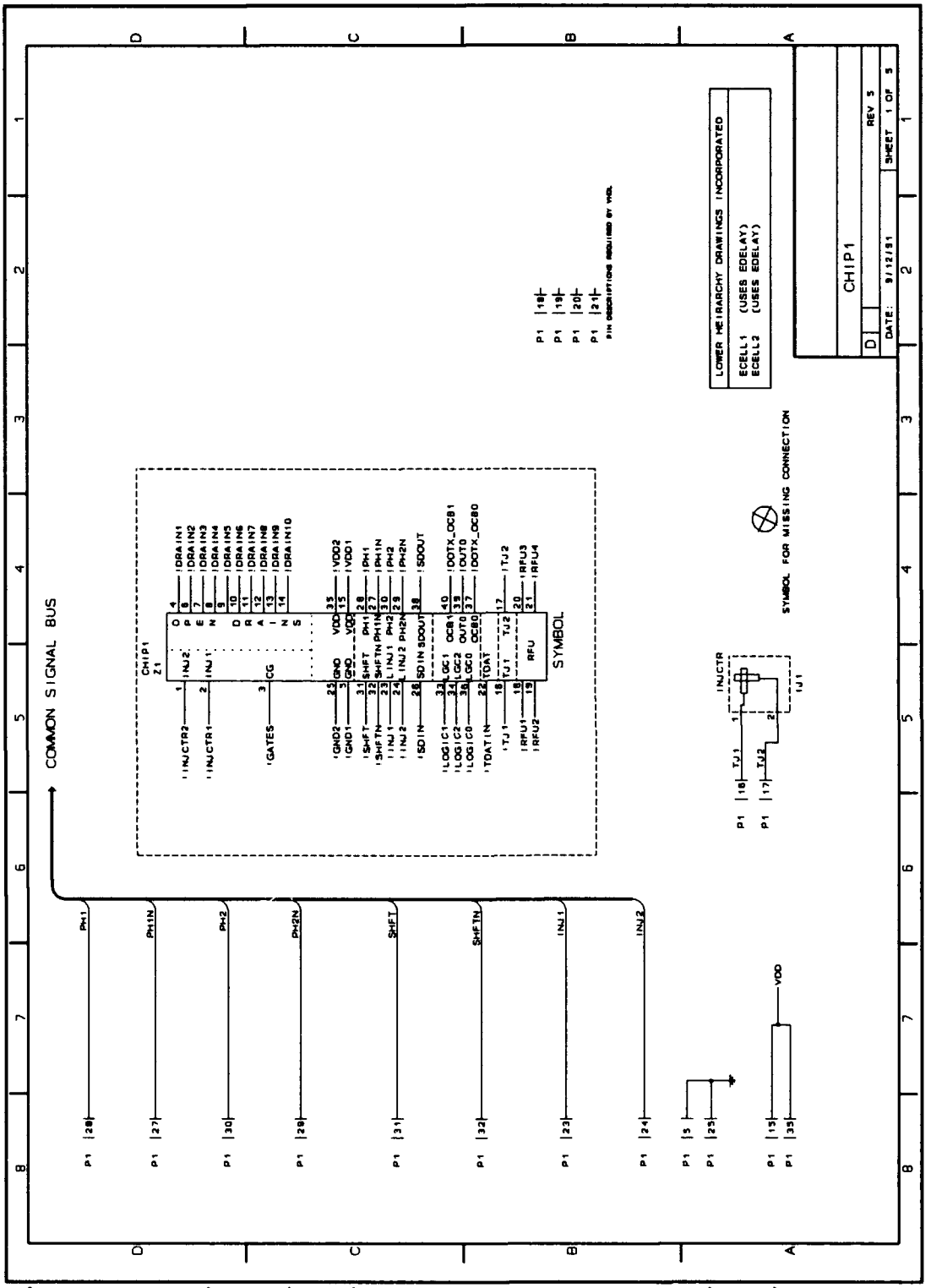


Figure 11. Microcircuit 1 Top Level Schematic Diagram. Sheet 1 of 5.

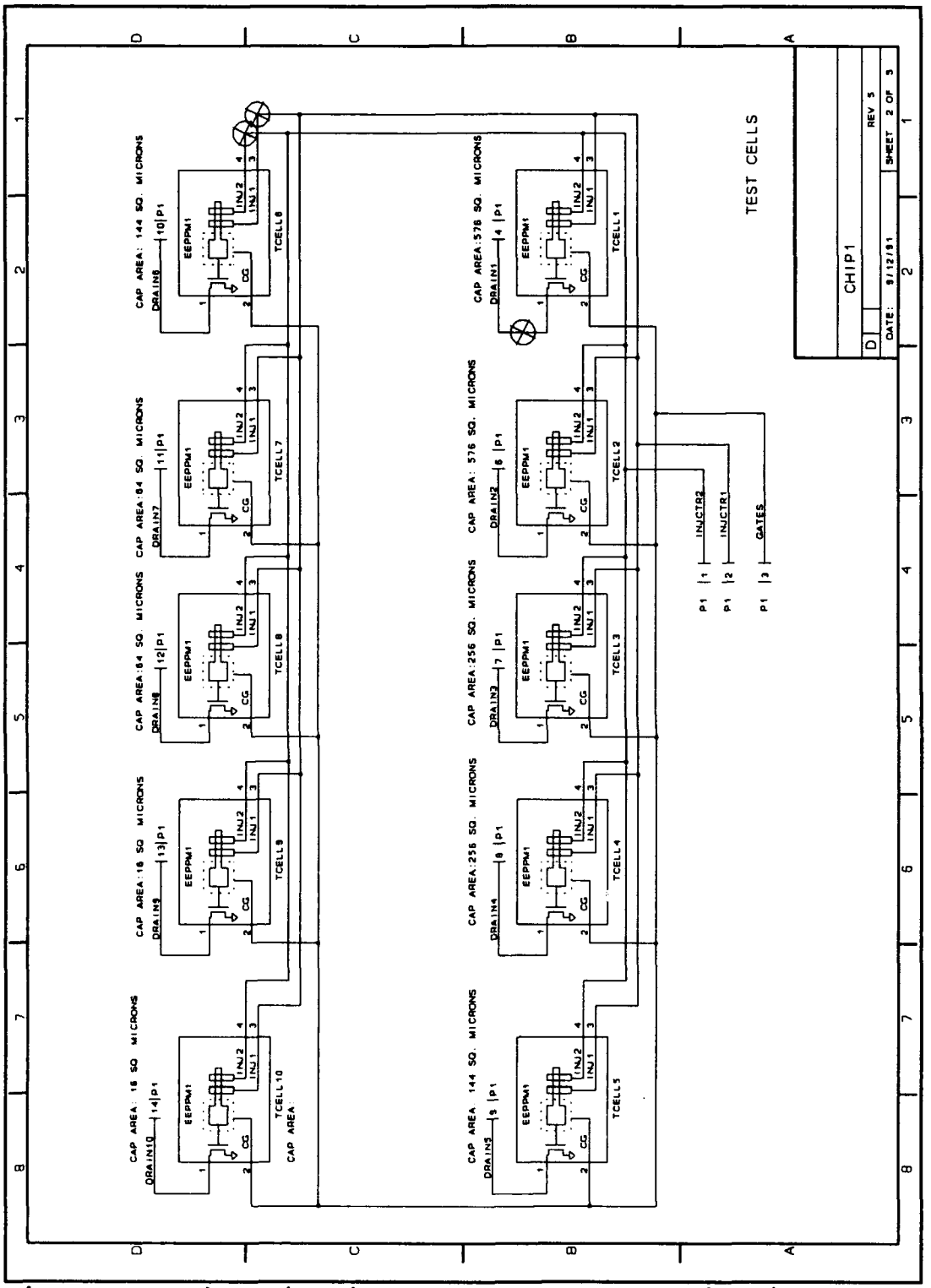


Figure 12. Microcircuit 1 Top Level Schematic Diagram. Sheet 2 of 5.

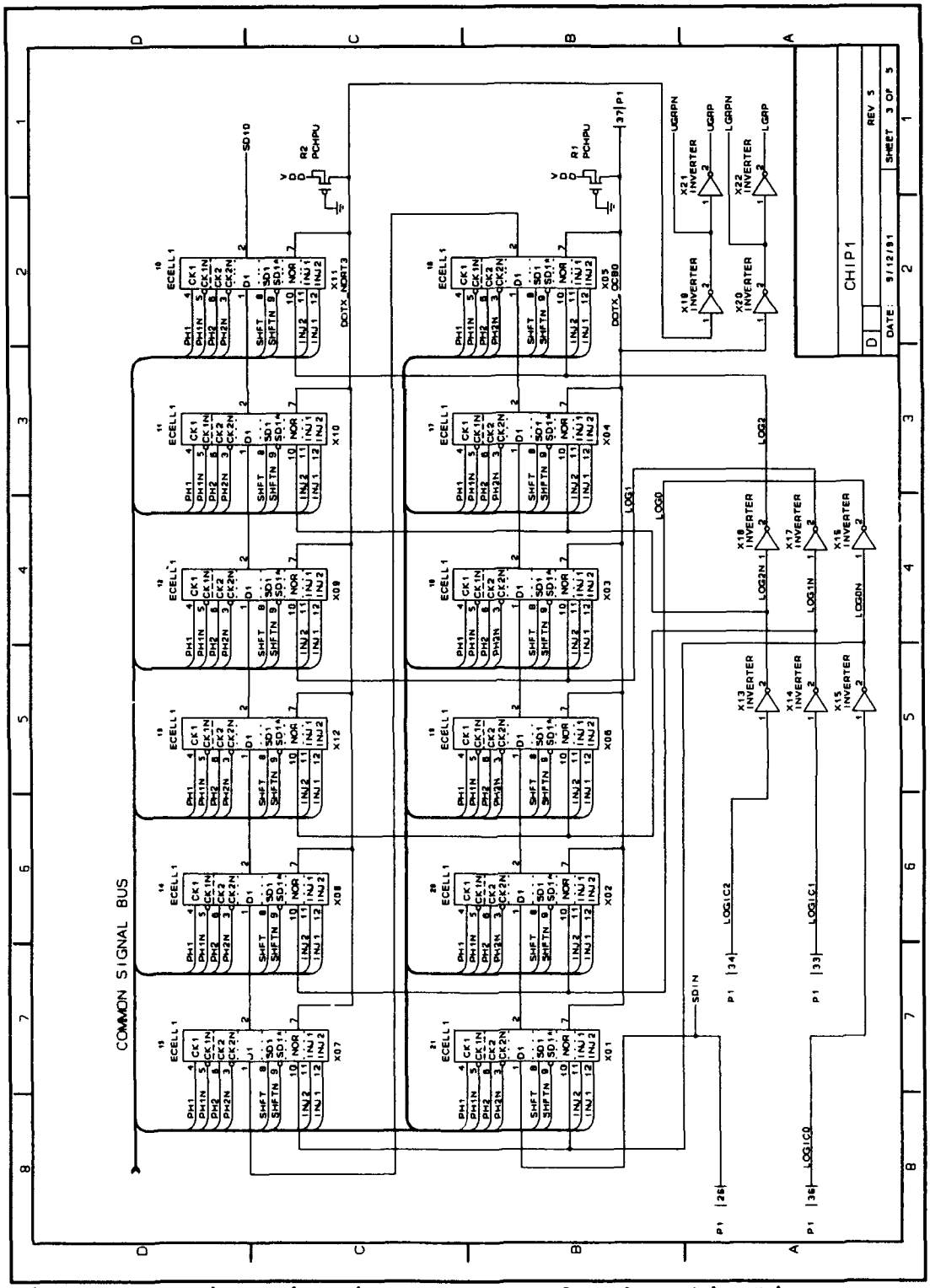


Figure 13. Microcircuit 1 Top Level Schematic Diagram. Sheet 3 of 5.

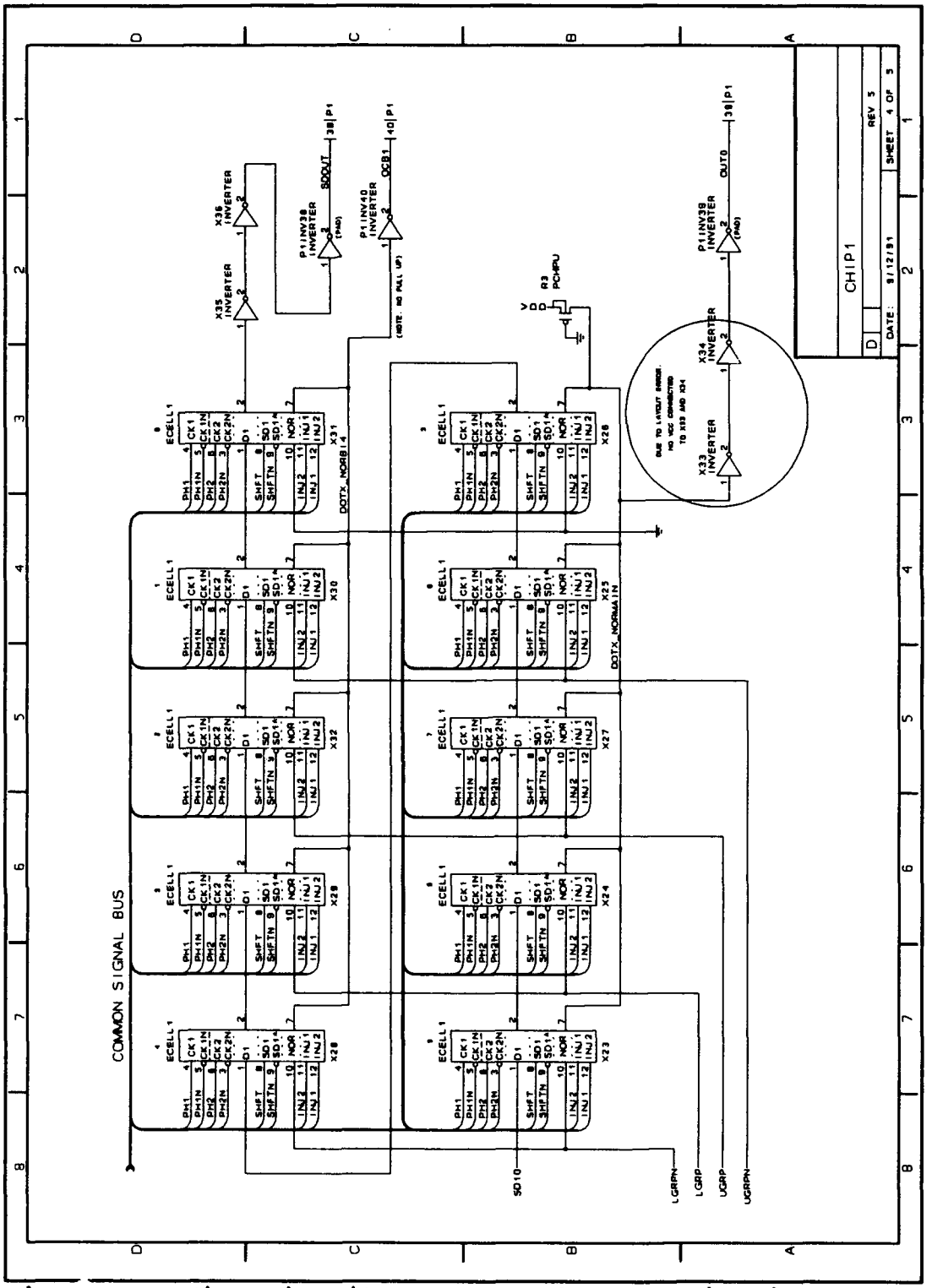


Figure 14. Microcircuit 1 Top Level Schematic Diagram. Sheet 4 of 5.

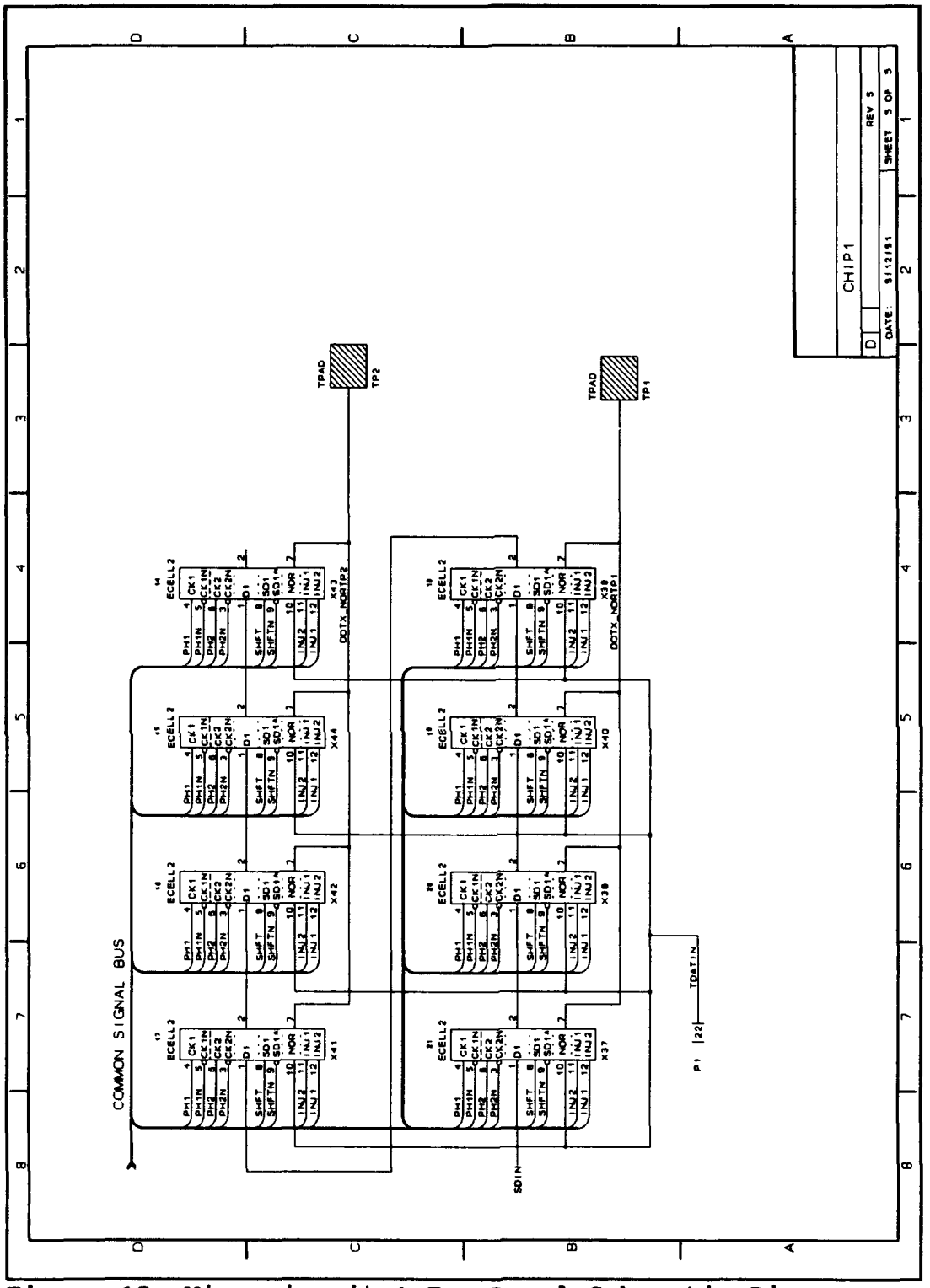


Figure 15. Microcircuit 1 Top Level Schematic Diagram. Sheet 5 of 5.

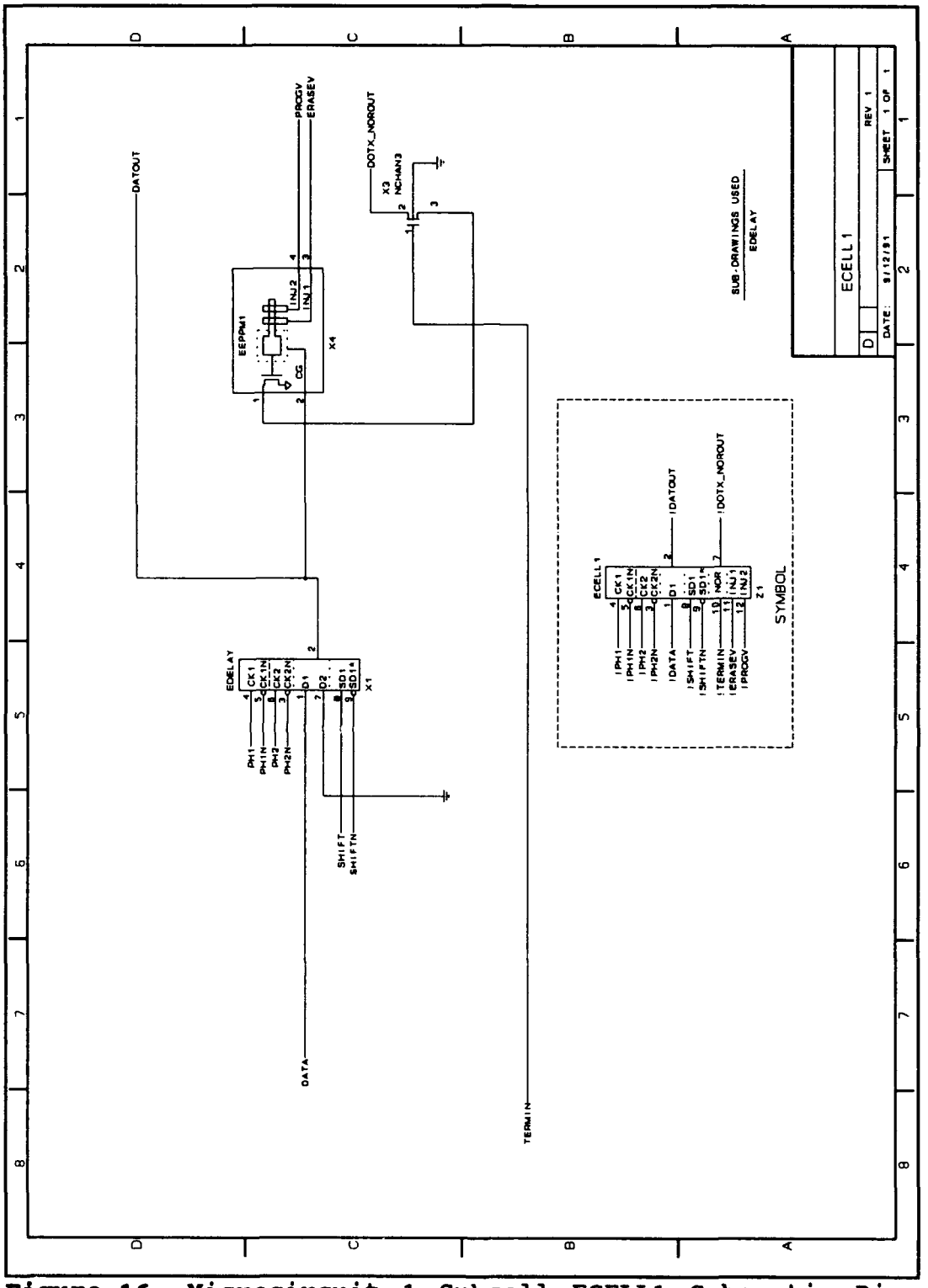


Figure 16. Microcircuit 1 Subcell ECELL1 Schematic Diagram.

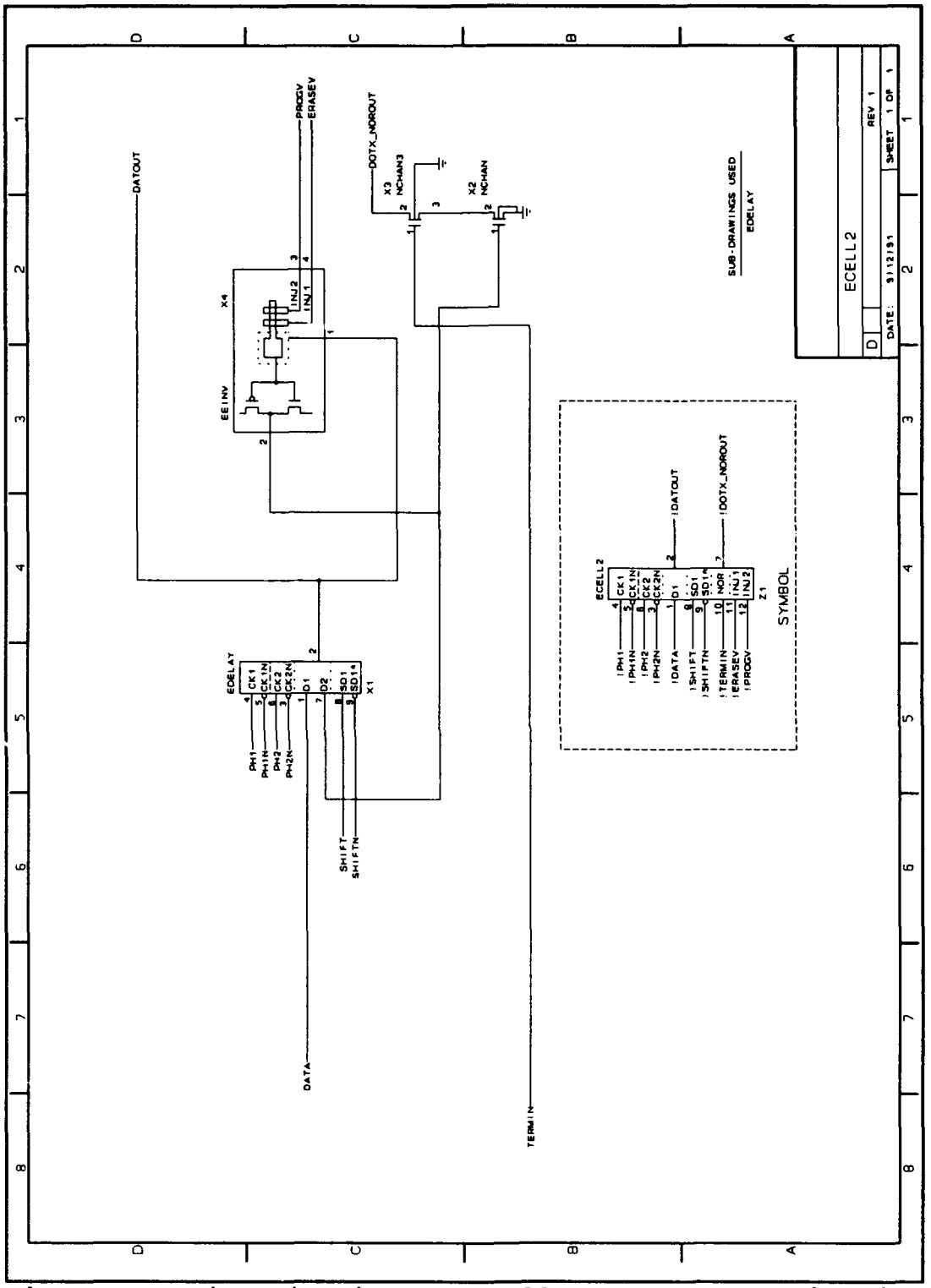


Figure 17. Microcircuit 1 Subcell ECELL2 Schematic Diagram.

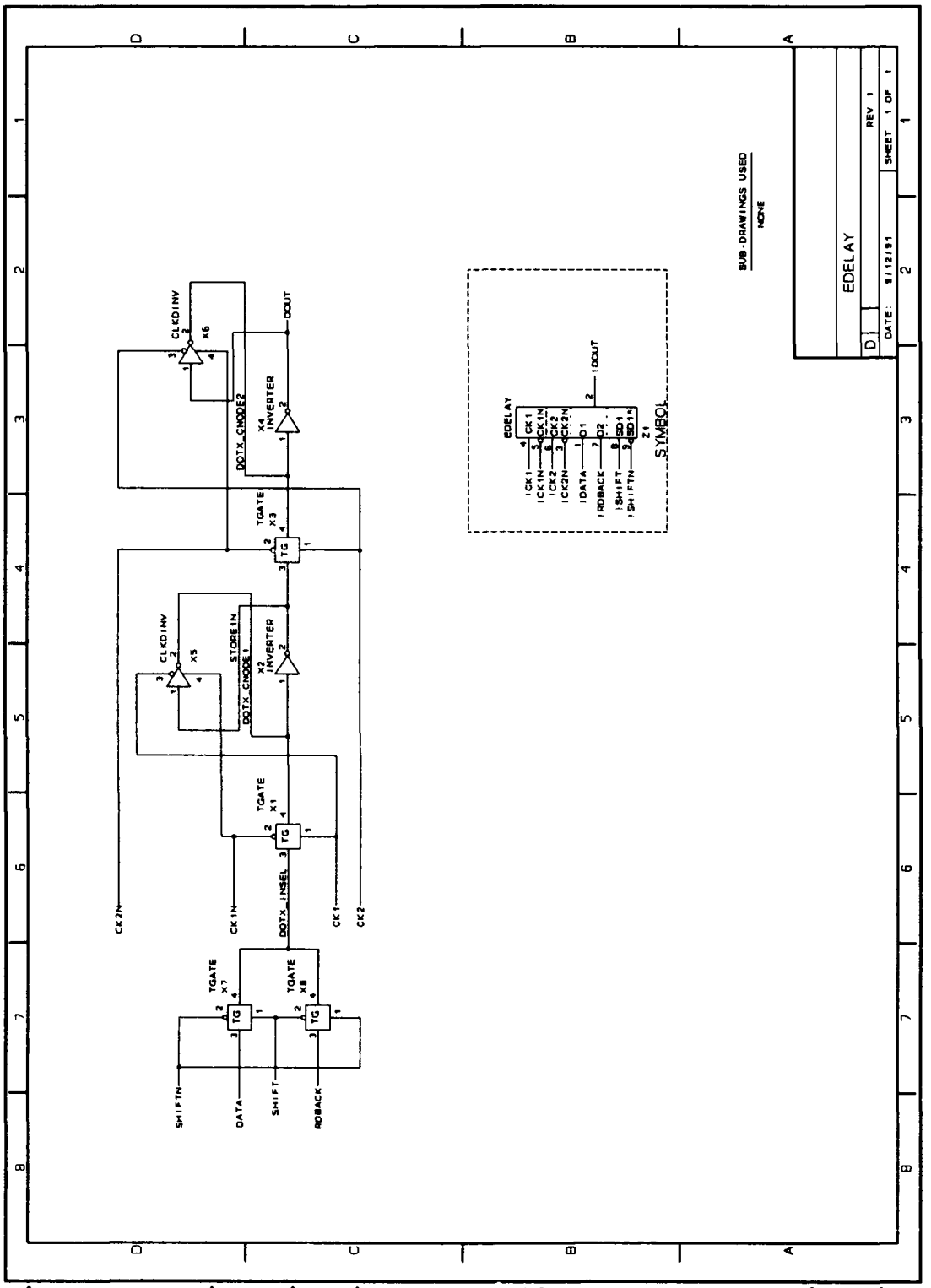


Figure 18. Microcircuit 1 Subcell EDELAY Schematic Diagram.

Appendix B: VHDL Models For Microcircuit 1.

Test Bench CH2BNCH VHDL Model

```
-----------------------------------------------------
-- Date:  19 Sep 91
-- Version:  2
--
-- Unix filename:  ch1bnch.vhd
--
-- This is a test bench for module CHIP1
-- The framework was generated by AVG, then customized
-- by adding the process and configuration sections
-----------------------------------------------------
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity CH1BNCH is
end CH1BNCH;
-- end entity

architecture A of CH1BNCH is

-- SIGNAL DECLARATIONS HERE
    signal GND : MVL7 :='0';
    signal VDD : MVL7 :='1';
    signal  CGATE : MVL7;
    signal  DRAIN1 : MVL7;
    signal  DRAIN2 : MVL7;
    signal  DRAIN3 : MVL7;
    signal  DRAIN4 : MVL7;
    signal  DRAIN5 : MVL7;
    signal  DRAIN6 : MVL7;
    signal  DRAIN7 : MVL7;
    signal  DRAIN8 : MVL7;
    signal  DRAIN9 : MVL7;
    signal  DRAIN10 : MVL7;
    signal  INJ1 : MVL7;
    signal  INJ2 : MVL7;
    signal  SDATIN : MVL7;
    signal  PH1N : MVL7;
    signal  PH1 : MVL7;
    signal  PH2N : MVL7;
    signal  PH2 : MVL7;
    signal  SHIFT : MVL7;
    signal  SHIFTN : MVL7;
    signal  LOGIC1 : MVL7;
    signal  LOGIC2 : MVL7;
    signal  LOGIC0 : MVL7;
    signal  OCB0 : DotX;
    signal  SDOUT : MVL7;
    signal  OUT0 : MVL7;
--       END SIGNALS

-- COMPONENT DECLARATIONS HERE
--
component CHIP1     port(
INJCTR2: in MVL7;
INJCTR1: in MVL7;
GATES: in MVL7;
DRAIN1: inout MVL7;
GND2: in MVL7;
DRAIN2: inout MVL7;
DRAIN3: inout MVL7;
DRAIN4: inout MVL7;
```

```
DRAIN5: inout MVL7;
DRAIN6: inout MVL7;
DRAIN7: inout MVL7;
DRAIN8: inout MVL7;
DRAIN9: inout MVL7;
DRAIN10: inout MVL7;
VDD1: in MVL7;
TJ1: in MVL7;
TJ2: in MVL7;
RFU1: in MVL7;
RFU2: in MVL7;
RFU3: in MVL7;
RFU4: in MVL7;
TDATIN: in MVL7;
INJ1: in MVL7;
INJ2: in MVL7;
GND1: in MVL7;
SDIN: in MVL7;
PH1N: in MVL7;
PH1: in MVL7;
PH2N: in MVL7;
PH2: in MVL7;
SHFT: in MVL7;
SHFTN: in MVL7;
LOGIC1: in MVL7;
LOGIC2: in MVL7;
VDD2: in MVL7;
LOGIC0: in MVL7;
OCB0: inout DotX;
SDOUT: out MVL7;
OUT0: out MVL7;
OCB1: inout DotX);
end component;

begin

X1: CHIP1 PORT MAP ( GND,GND, CGATE, DRAIN1, GND,
                     DRAIN2, DRAIN3, DRAIN4, DRAIN5, DRAIN6,
                     DRAIN7, DRAIN8, DRAIN9, DRAIN10, VDD,
                     GND,GND,GND,GND,GND,
                     GND,GND, INJ1, INJ2, GND,
                     SDATIN, PH1N, PH1, PH2N, PH2,
                     SHIFT, SHIFTN, LOGIC1, LOGIC2, VDD,
                     LOGIC0, OCB0, SDOUT, OUT0, OPEN);

DRIVE: Process
procedure tick is
begin
PH1 <= '1';
PH1N <= '0';
wait for 20 ns;
PH1 <= '0';
PH1N <= '1';
PH2 <= '1';
PH2N <= '0';
wait for 20 NS;
PH2 <= '0';
PH2N <= '1';
wait for 20 NS;
end tick;

procedure ERASE is
begin
-- ERASE CYCLE
SDATIN <= '1';
PH1 <= '1';
PH2 <= '1';
```

```
PH1N <= '0';
PH2N <= '0';
-- BOTH CLOCKS TRUE. ALL GATES ARE 1.
wait for 250 NS;
INJ2 <= '1';
wait for 50 NS;
INJ2 <= '0';
PH1 <= '0';
PH2 <= '0';
PH1N <= '1';
PH2N <= '1';
wait for 50 NS;
-- ALL CELLS SHOULD NOW BE ERASED
end ERASE;

procedure SRWRITE ( dat : in integer) is
variable temvar,temdat : integer;
variable temvas: MVL7;
-- Will load 22 bit shift register with data pattern in dat
begin
  temdat := dat;
Loop1:
-- This loop takes the input data and shifts the
-- complement into the register
  for indx in 0 to 21 loop
    temvar:= temdat rem 2;
    temdat := temdat / 2;
    if temvar = 1 then temvas := '0';
      else temvas := '1';
    end if;
    SDATIN <= temvas;
    tick;
  end loop loop1;
end SRWRITE;

procedure PROGRAM ( dat : in integer) is
variable temvar,temdat : integer;
begin
ERASE;
SRWRITE(dat);
INJ1 <= '1';
WAIT FOR 20 NS;
INJ1 <= '0';
-- Cells are programmed now
-- logic test now
WAIT FOR 10 NS;
-- CASE 000 (i.e. LOGIC2 = 0, LOGIC1 = 0, LOGIC0 = 0
LOGIC0 <= '0';
LOGIC1 <= '0';
LOGIC2 <= '0';
WAIT FOR 20 NS;
-- CASE 001
LOGIC0 <= '1';
WAIT FOR 10 NS;
-- CASE 010
LOGIC0 <= '0';
LOGIC1 <= '1';
WAIT FOR 10 NS;
--  CASE 011
LOGIC0 <= '1';
WAIT FOR 10 NS;
-- CASE 100
LOGIC0 <= '0';
LOGIC1 <= '0';
LOGIC2 <= '1';
WAIT FOR 20 NS;
-- CASE 101
```

```
LOGIC0 <= '1';
WAIT FOR 20 NS;
-- CASE 110
LOGIC0 <= '0';
LOGIC1 <= '1';
WAIT FOR 20 NS;
--  CASE 111
LOGIC0 <= '1';
WAIT FOR 20 NS;
end PROGRAM;


-- BEGIN TEST
begin
-- INITIALIZE
INJ1 <= '0';
INJ2 <= '0';
SDATIN <= '0';
PH1 <= '0';
PH1N <= '1';
PH2 <= '0';
PH2N <= '1';
LOGIC0 <= '1';
LOGIC1 <= '1';
LOGIC2 <= '1';
SHIFT <= '1';
SHIFTN <= '0';
WAIT FOR 10 NS;
PROGRAM(16#0A840#);
PROGRAM(16#05480#);
PROGRAM(16#0A880#);
PROGRAM(16#05440#);
PROGRAM(16#246180#);
PROGRAM(16#285180#);
-- Now the OCB0 tests
PROGRAM(16#2A0000#);
PROGRAM(16#150000#);
WAIT;
end process;
end A;

configuration CTEST of CH1BNCH is
   for A
         for all : CHIP1
                        use entity WORK.CHIP1(structure);
         end for;
      end for;
end CTEST;
```

Top Level CHIP1 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE CHIP1(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental)
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--      Modified to remove connector P1 references
--     Generated on 09-19-1991 at 16:46:28
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity CHIP1 is
     port(
INJCTR2: in MVL7;
INJCTR1: in MVL7;
GATES: in MVL7;
DRAIN1: inout MVL7;
GND2: in MVL7;
DRAIN2: inout MVL7;
DRAIN3: inout MVL7;
DRAIN4: inout MVL7;
DRAIN5: inout MVL7;
DRAIN6: inout MVL7;
DRAIN7: inout MVL7;
DRAIN8: inout MVL7;
DRAIN9: inout MVL7;
DRAIN10: inout MVL7;
VDD1: in MVL7;
TJ1: in MVL7;
TJ2: in MVL7;
RFU1: in MVL7;
RFU2: in MVL7;
RFU3: in MVL7;
RFU4: in MVL7;
TDATIN: in MVL7;
INJ1: in MVL7;
INJ2: in MVL7;
GND1: in MVL7;
SDIN: in MVL7;
PH1N: in MVL7;
PH1: in MVL7;
PH2N: in MVL7;
PH2: in MVL7;
SHFT: in MVL7;
SHFTN: in MVL7;
LOGIC1: in MVL7;
LOGIC2: in MVL7;
VDD2: in MVL7;
LOGIC0: in MVL7;
OCB0: inout DotX;
SDOUT: out MVL7;
OUT0: out MVL7;
OCB1: inout DotX );
end CHIP1;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of CHIP1 is

-- SIGNAL DECLARATIONS HERE
        signal GND : MVL7 :='0';
        signal VDD : MVL7 :='1';
    signal  S04076022 : MVL7;
    signal  S04075060 : MVL7;
    signal  NORBI4 : DotX;
```

```
    signal  NORT3 : DotX;
    signal  NORMAIN : DotX;
    signal  NORTP1 : DotX;
    signal  NORTP2 : DotX;
    signal  S03020042 : MVL7;
    signal  LOGON : MVL7;
    signal  S03033042 : MVL7;
    signal  LOG0 : MVL7;
    signal  S03045042 : MVL7;
    signal  S03057042 : MVL7;
    signal  LOG1 : MVL7;
    signal  S03070042 : MVL7;
    signal  LOG2N : MVL7;
    signal  S03007020 : MVL7;
    signal  LOG2 : MVL7;
    signal  LOG1N : MVL7;
    signal  S03020020 : MVL7;
    signal  S03033020 : MVL7;
    signal  S03045020 : MVL7;
    signal  S03057020 : MVL7;
    signal  S03070020 : MVL7;
    signal  SD10 : MVL7;
    signal  UGRPN : MVL7;
    signal  LGRPN : MVL7;
    signal  UGRP : MVL7;
    signal  LGRP : MVL7;
    signal  S04020042 : MVL7;
    signal  S04033042 : MVL7;
    signal  S04045042 : MVL7;
    signal  S04057042 : MVL7;
    signal  S04007020 : MVL7;
    signal  S04020020 : MVL7;
    signal  S04033020 : MVL7;
    signal  S04045020 : MVL7;
    signal  S04057020 : MVL7;
    signal  S04070020 : MVL7;
    signal  S04066060 : MVL7;
    signal  S04079020 : MVL7;
    signal  S05020042 : MVL7;
    signal  S05033042 : MVL7;
    signal  S05045042 : MVL7;
    signal  S05007020 : MVL7;
    signal  S05020020 : MVL7;
    signal  S05033020 : MVL7;
    signal  S05045020 : MVL7;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

component INJCTR     port(
PIN1 : in MVL7;
PIN2 : in MVL7);
end component;


component inverter
  port(input: in MVL7; output: out MVL7);
end component;
component pchpu
-- P-CHANNEL MOS TRANSISTOR USED AS A PULL-UP RESISTOR
  port( PULLUP  : inout DotX);
end component;
component EEPPM1
  port(DRAIN : inout DotX;
       GATE : in MVL7;
       INJ1 : in MVL7;
```

```
      INJ2 : in MVL7);
end component;
component TPAD     port(
PIN1 : in MVL7);
end component;

-- COMPONENT GENERATED BY AVG FROM FILE ECELL1(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental)
--            Generated on 09-19-1991 at 16:46:26
--
component ECELL1     port(
DATA: in MVL7;
DATOUT: inout MVL7;
PH2N: in MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
NOROUT: inout DotX;
SHIFT: in MVL7;
SHIFTN: in MVL7;
TERMIN: in MVL7;
ERASEV: in MVL7;
PROGV: in MVL7);
end component;

-- COMPONENT GENERATED BY AVG FROM FILE ECELL2(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental)
--            Generated on 09-19-1991 at 16:46:27
--
component ECELL2     port(
DATA: in MVL7;
DATOUT: inout MVL7;
PH2N: in MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
NOROUT: inout DotX;
SHIFT: in MVL7;
SHIFTN: in MVL7;
TERMIN: in MVL7;
ERASEV: in MVL7;
PROGV: in MVL7);
end component;


begin


IJ1: INJCTR PORT MAP ( TJ1, TJ2);


P1INV38: INVERTER PORT MAP ( S04076022, SDOUT);

P1INV39: INVERTER PORT MAP ( S04075060, OUT0);

P1INV40: INVERTER PORT MAP ( NORBI4, OCB1);

R1: PCHPU PORT MAP ( OCB0);

R2: PCHPU PORT MAP ( NORT3);

R3: PCHPU PORT MAP ( NORMAIN);

TCELL1: EEPPM1 PORT MAP ( DRAIN1, GATES, INJCTR1, INJCTR2);

TCELL10: EEPPM1 PORT MAP ( DRAIN10, GATES, INJCTR1, INJCTR2);
```

```
TCELL2: EEPPM1 PORT MAP ( DRAIN2, GATES, INJCTR1, INJCTR2);

TCELL3: EEPPM1 PORT MAP ( DRAIN3, GATES, INJCTR1, INJCTR2);

TCELL4: EEPPM1 PORT MAP ( DRAIN4, GATES, INJCTR1, INJCTR2);

TCELL5: EEPPM1 PORT MAP ( DRAIN5, GATES, INJCTR1, INJCTR2);

TCELL6: EEPPM1 PORT MAP ( DRAIN6, GATES, INJCTR1, INJCTR2);

TCELL7: EEPPM1 PORT MAP ( DRAIN7, GATES, INJCTR1, INJCTR2);

TCELL8: EEPPM1 PORT MAP ( DRAIN8, GATES, INJCTR1, INJCTR2);

TCELL9: EEPPM1 PORT MAP ( DRAIN9, GATES, INJCTR1, INJCTR2);

TP1: TPAD PORT MAP ( NORTP1);

TP2: TPAD PORT MAP ( NORTP2);

X01: ECELL1 PORT MAP ( SDIN, S03020042, PH2N, PH1, PH1N,
                       PH2, OCBO, SHFT, SHFTN, LOGON,
                       INJ2, INJ1);

X02: ECELL1 PORT MAP ( S03020042, S03033042, PH2N, PH1, PH1N,
                       PH2, OCBO, SHFT, SHFTN, LOGO,
                       INJ2, INJ1);

X03: ECELL1 PORT MAP ( S03045042, S03057042, PH2N, PH1, PH1N,
                       PH2, OCBO, SHFT, SHFTN, LOG1,
                       INJ2, INJ1);

X04: ECELL1 PORT MAP ( S03057042, S03070042, PH2N, PH1, PH1N,
                       PH2, OCBO, SHFT, SHFTN, LOG2N,
                       INJ2, INJ1);

X05: ECELL1 PORT MAP ( S03070042, S03007020, PH2N, PH1, PH1N,
                       PH2, OCBO, SHFT, SHFTN, LOG2,
                       INJ2, INJ1);

X06: ECELL1 PORT MAP ( S03033042, S03045042, PH2N, PH1, PH1N,
                       PH2, OCBO, SHFT, SHFTN, LOG1N,
                       INJ2, INJ1);

X07: ECELL1 PORT MAP ( S03007020, S03020020, PH2N, PH1, PH1N,
                       PH2, NORT3, SHFT, SHFTN, LOGON,
                       INJ2, INJ1);

X08: ECELL1 PORT MAP ( S03020020, S03033020, PH2N, PH1, PH1N,
                       PH2, NORT3, SHFT, SHFTN, LOGO,
                       INJ2, INJ1);

X09: ECELL1 PORT MAP ( S03045020, S03057020, PH2N, PH1, PH1N,
                       PH2, NORT3, SHFT, SHFTN, LOG1,
                       INJ2, INJ1);

X10: ECELL1 PORT MAP ( S03057020, S03070020, PH2N, PH1, PH1N,
                       PH2, NORT3, SHFT, SHFTN, LOG2N,
                       INJ2, INJ1);

X11: ECELL1 PORT MAP ( S03070020, SD10, PH2N, PH1, PH1N,
                       PH2, NORT3, SHFT, SHFTN, LOG2,
                       INJ2, INJ1);

X12: ECELL1 PORT MAP ( S03033020, S03045020, PH2N, PH1, PH1N,
                       PH2, NORT3, SHFT, SHFTN, LOG1N,
                       INJ2, INJ1);
```

```
X13: INVERTER PORT MAP ( LOGIC2, LOG2N);

X14: INVERTER PORT MAP ( LOGIC1, LOG1N);

X15: INVERTER PORT MAP ( LOGIC0, LOG0N);

X16: INVERTER PORT MAP ( LOG0N, LOG0);

X17: INVERTER PORT MAP ( LOG1N, LOG1);

X18: INVERTER PORT MAP ( LOG2N, LOG2);

X19: INVERTER PORT MAP ( NORT3, UGRPN);

X20: INVERTER PORT MAP ( OCB0, LGRPN);

X21: INVERTER PORT MAP ( UGRPN, UGRP);

X22: INVERTER PORT MAP ( LGRPN, LGRP);

X23: ECELL1 PORT MAP ( SD10, S04020042, PH2N, PH1, PH1N,
                       PH2, NORMAIN, SHFT, SHFTN, LGRPN,
                       INJ2, INJ1);

X24: ECELL1 PORT MAP ( S04020042, S04033042, PH2N, PH1, PH1N,
                       PH2, NORMAIN, SHFT, SHFTN, LGRP,
                       INJ2, INJ1);

X25: ECELL1 PORT MAP ( S04045042, S04057042, PH2N, PH1, PH1N,
                       PH2, NORMAIN, SHFT, SHFTN, UGRPN,
                       INJ2, INJ1);

X26: ECELL1 PORT MAP ( S04057042, S04007020, PH2N, PH1, PH1N,
                       PH2, NORMAIN, SHFT, SHFTN, GND,
                       INJ2, INJ1);

X27: ECELL1 PORT MAP ( S04033042, S04045042, PH2N, PH1, PH1N,
                       PH2, NORMAIN, SHFT, SHFTN, UGRP,
                       INJ2, INJ1);

X28: ECELL1 PORT MAP ( S04007020, S04020020, PH2N, PH1, PH1N,
                       PH2, NORBI4, SHFT, SHFTN, LGRPN,
                       INJ2, INJ1);

X29: ECELL1 PORT MAP ( S04020020, S04033020, PH2N, PH1, PH1N,
                       PH2, NORBI4, SHFT, SHFTN, LGRP,
                       INJ2, INJ1);

X30: ECELL1 PORT MAP ( S04045020, S04057020, PH2N, PH1, PH1N,
                       PH2, NORBI4, SHFT, SHFTN, UGRPN,
                       INJ2, INJ1);

X31: ECELL1 PORT MAP ( S04057020, S04070020, PH2N, PH1, PH1N,
                       PH2, NORBI4, SHFT, SHFTN, GND,
                       INJ2, INJ1);

X32: ECELL1 PORT MAP ( S04033020, S04045020, PH2N, PH1, PH1N,
                       PH2, NORBI4, SHFT, SHFTN, UGRP,
                       INJ2, INJ1);

X33: INVERTER PORT MAP ( NORMAIN, S04066060);

X34: INVERTER PORT MAP ( S04066060, S04075060);

X35: INVERTER PORT MAP ( S04070020, S04079020);

X36: INVERTER PORT MAP ( S04079020, S04076022);
```

```
X37: ECELL2 PORT MAP ( SDIN, S05020042, PH2N, PH1, PH1N,
                       PH2, NORTP1, SHFT, SHFTN, TDATIN,
                       INJ2, INJ1);

X38: ECELL2 PORT MAP ( S05020042, S05033042, PH2N, PH1, PH1N,
                       PH2, NORTP1, SHFT, SHFTN, TDATIN,
                       INJ2, INJ1);

X39: ECELL2 PORT MAP ( S05045042, S05007020, PH2N, PH1, PH1N,
                       PH2, NORTP1, SHFT, SHFTN, TDATIN,
                       INJ2, INJ1);

X40: ECELL2 PORT MAP ( S05033042, S05045042, PH2N, PH1, PH1N,
                       PH2, NORTP1, SHFT, SHFTN, TDATIN,
                       INJ2, INJ1);

X41: ECELL2 PORT MAP ( S05007020, S05020020, PH2N, PH1, PH1N,
                       PH2, NORTP2, SHFT, SHFTN, TDATIN,
                       INJ2, INJ1);

X42: ECELL2 PORT MAP ( S05020020, S05033020, PH2N, PH1, PH1N,
                       PH2, NORTP2, SHFT, SHFTN, TDATIN,
                       INJ2, INJ1);

X43: ECELL2 PORT MAP ( S05045020, OPEN, PH2N, PH1, PH1N,
                       PH2, NORTP2, SHFT, SHFTN, TDATIN,
                       INJ2, INJ1);

X44: ECELL2 PORT MAP ( S05033020, S05045020, PH2N, PH1, PH1N,
                       PH2, NORTP2, SHFT, SHFTN, TDATIN,
                       INJ2, INJ1);

-- END OF AUTOMATICALLY GENERATED VHDL SiRUCTURE

end structure ;
```

Subcell ECELL1 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ECELL1(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 09-19-1991 at 16:46:26
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ECELL1 is
     port(
DATA: in MVL7;
DATOUT: inout MVL7;
PH2N: in MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
NOROUT: inout DotX;
SHIFT: in MVL7;
SHIFTN: in MVL7;
TERMIN: in MVL7;
ERASEV: in MVL7;
PROGV: in MVL7 );
end ECELL1;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ECELL1 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  S01067021 : MVL7;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE EDELAY(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 09-19-1991 at 16:46:25
--
component EDELAY     port(
DATA: in MVL7;
DOUT: inout MVL7;
CK2N: in MVL7;
CK1: in MVL7;
CK1N: in MVL7;
CK2: in MVL7;
RDBACK: in MVL7;
SHIFT: in MVL7;
SHIFTN: in MVL7);
end component;

component nchan3
  port(Gate: in MVL7;
       Drain: out DotX;
       Source: in MVL7);
end component;
component EEPPM1
  port(DRAIN : inout DotX;
       GATE : in MVL7;
       INJ1 : in MVL7;
```

```
      INJ2 : in MVL7);
end component;

begin


X1: EDELAY PORT MAP ( DATA, DATOUT, PH2N, PH1, PH1N,
                      PH2, GND, SHIFT, SHIFTN);

X3: NCHAN3 PORT MAP ( TERMIN, NOROUT, S01067021);

X4: EEPPM1 PORT MAP ( S01067021, DATOUT, ERASEV, PROGV);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ECELL2 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ECELL2(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental)
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 09-19-1991 at 16:46:27
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ECELL2 is
     port(
DATA: in MVL7;
DATOUT: inout MVL7;
PH2N: in MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
NOROUT: inout DotX;
SHIFT: in MVL7;
SHIFTN: in MVL7;
TERMIN: in MVL7;
ERASEV: in MVL7;
PROGV: in MVL7 );
end ECELL2;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ECELL2 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  S01037026 : MVL7;
    signal  S01084037 : MVL7;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE
-- COMPONENT GENERATED BY AVG FROM FILE EDELAY(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental)
--           Generated on 09-19-1991 at 16:46:25
--
component EDELAY     port(
DATA: in MVL7;
DOUT: inout MVL7;
CK2N: in MVL7;
CK1: in MVL7;
CK1N: in MVL7;
CK2: in MVL7;
RDBACK: in MVL7;
SHIFT: in MVL7;
SHIFTN: in MVL7);
end component;

component nchan
  port(Gate: in MVL7; Drain: out MVL7);
end component;
component nchan3
  port(Gate: in MVL7;
       Drain: out DotX;
       Source: in MVL7);
end component;
component EEINV
```

```
  port(GATE : in MVL7;
       OUT1 : out MVL7;
       INJ2 : in MVL7;
       INJ1 : in MVL7);
end component;

begin

X1: EDELAY PORT MAP ( DATA, DATOUT, PH2N, PH1, PH1N,
                      PH2, S01037026, SHIFT, SHIFTN);

X2: NCHAN PORT MAP ( S01037026, S01084037);

X3: NCHAN3 PORT MAP ( TERMIN, NOROUT, S01084037);

X4: EEINV PORT MAP ( DATOUT, S01037026, PROGV, ERASEV);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell EDELAY VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE EDELAY(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental)
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 09-19-1991 at 16:46:25
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity EDELAY is
     port(
DATA: in MVL7;
DOUT: inout MVL7;
CK2N: in MVL7;
CK1: in MVL7;
CK1N: in MVL7;
CK2: in MVL7;
RDBACK: in MVL7;
SHIFT: in MVL7;
SHIFTN: in MVL7 );
end EDELAY;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of EDELAY is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  INSEL : DotX;
    signal  CNODE1 : DotX;
    signal  STORE1N : MVL7;
    signal  CNODE2 : DotX;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

component tgate
  port(p1 : in MVL7;
       p2 : in MVL7;
       g : in MVL7;
       d  : inout DotX);
end component;
component inverter
  port(input: in MVL7; output: out MVL7);
end component;
component clkdinv
  port(INPUT : in MVL7;
       OUTPUT  : inout DotX;
       PH1  : in MVL7;
       PH1N : in MVL7);
end component;

begin


X1: TGATE PORT MAP ( CK1, CK1N, INSEL, CNODE1);

X2: INVERTER PORT MAP ( CNODE1, STORE1N);

X3: TGATE PORT MAP ( CK2, CK2N, STORE1N, CNODE2);
```

```
X4: INVERTER PORT MAP ( CNODE2, DOUT);
X5: CLKDINV PORT MAP ( STORE1N, CNODE1, CK1N, CK1);
X6: CLKDINV PORT MAP ( DOUT, CNODE2, CK2N, CK2);
X7: TGATE PORT MAP ( SHIFT, SHIFTN, DATA, INSEL);
X8: TGATE PORT MAP ( SHIFTN, SHIFT, RDBACK, INSEL);
-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE
end structure ;
```

Subcell EEPPM1 VHDL Model

```
-------------------------------------------------------
-- Date:  14 Sep 91
-- Version:  3
--
-- Unix filename:  eeppm1.vhd
--
-- Function:  This file is a behavioral description of an
-- EEPROM cell with dual injectors. For simplicity, injector 1
-- is assumed (and required) to be pulsed for programming
-- and injector 2 is pulsed for erase. Standard MVL7 inputs to
-- the injectors are used, rather than the actual high voltage
-- positive and negative pulses. This allows logical verification.
-- This cell has an open drain output. The drain output
-- is not a resolved wire since it will feed only one input.
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity EEPPM1 is
  port(DRAIN : inout DotX;
       GATE : in MVL7;
       INJ1 : in MVL7;
       INJ2 : in MVL7);
end EEPPM1;

architecture behavioral of EEPPM1 is

Signal Floating_Gate: MVL7;
begin
process
variable temp_output: MVL7;
begin

-- Program Cell
    if MVL7toBIT(Gate) = '0' and MVL7toBIT(INJ2) = '1'
       then Floating_Gate <= '1';
    end if;
-- Erase Cell
    if MVL7toBIT(Gate) = '1' and MVL7toBIT(INJ1) = '1'
       then Floating_Gate <= '0';
    end if;
-- output is zero if floating gate is programmed
   if MVL7toBIT(Floating_Gate) = '1' then temp_output := '0';
    else temp_output := 'Z';
    end if;
    Drain <= temp_output;
    wait on INJ1, INJ2, Gate;
end process;
end behavioral;
```

Subcell EEINV VHDL Model

```
-------------------------------------------------------
-- Date:  14 Sep 91
-- Version:  3
--
-- Unix filename:  eeinv.vhd
--
-- Function:  This file is a behavioral description of an
-- EEPROM cell with dual injectors. For simplicity, injector 1
-- is assumed (and required) to be pulsed for programming
-- and injector 2 is pulsed for erase. Standard MVL7 inputs to
-- the injectors are used, rather than the actual high voltage
-- positive and negative pulses. This allows logical verification.
-- This cell has an active output (programmed inverter)
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity EEINV is
  port(GATE : in MVL7;
       OUT1 : out MVL7;
       INJ2 : in MVL7;
       INJ1 : in MVL7);
end EEINV;


architecture behavioral of EEINV is

Signal Floating_Gate: MVL7;
begin
process
variable temp_output: MVL7;
begin

-- Program Cell
    if MVL7toBIT(Gate) = '0' and MVL7toBIT(INJ1) = '1'
       then Floating_Gate <= '1';
    end if;
-- Erase Cell
    if MVL7toBIT(Gate) = '1' and MVL7toBIT(INJ2) = '1'
       then Floating_Gate <= '0';
    end if;
-- output is zero if floating gate is programmed
   if MVL7toBIT(Floating_Gate) = '1' then temp_output := '0';
    else temp_output := '1';
    end if;
    OUT1 <= temp_output;
    wait on INJ1, INJ2, Gate;
end process;
end behavioral;
```

Subcell CLKDINV VHDL Model

```
-------------------------------------------------------
-- Date:  14 Sep 91
-- Version:  3
--
-- Unix filename:  clkdinv.vhd
--
-- Function:  Dummy
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity clkdinv is
  port(INPUT : in MVL7;
       OUTPUT  : inout DotX;
       PH1  : in MVL7;
       PH1N : in MVL7);
end clkdinv;

architecture behavioral of clkdinv is
begin
process
begin
-- dummy
OUTPUT <= 'Z';
wait on input, ph1, ph1n;
end process;
end behavioral;
```

Subcell INJ VHDL Model

```
-------------------------------------------------------
-- Date:  14 Sep 91
-- Version:  3
--
-- Unix filename:  injctr.vhd
--
-- Function:  Dummy to allow analysis.
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity Injctr is    port(
PIN1 : in MVL7;
PIN2 : in MVL7);
end Injctr;

architecture behavioral of Injctr is

begin
process
begin
--dummy
wait;
end process;
end behavioral;
```

Subcell INVERTER VHDL Model

```
-------------------------------------------------------
-- Date:  12 Feb 91
-- Version:  5
--
-- Unix filename:  inverter.vhd
--
-- Function:  This file is a behavioral description of an inverter.
--  Contains both the entity and behavioral architecture of the
--  inverter. (MVL7 version).
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity inverter is
  port(input: in MVL7; output: out MVL7);
end inverter;

architecture behavioral of inverter is

begin
process

variable temp_output: BIT;

begin
    if MVL7toBIT(input) = '1' then temp_output := '0';
    else temp_output := '1';
    end if;
    output <= BITtoMVL7(temp_output) after 1 NS;
    wait on input;
end process;
end behavioral;
```

Subcell NCHAN VHDL Model

```
-------------------------------------------------------
-- Date:  12 Feb 91
-- Version:  1
--
-- Unix filename:  nchan.vhd
--
-- Function:  This file is a behavioral description of an
-- N-channel transistor connected as an open drain inverter.
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity nchan is
  port(Gate: in MVL7; Drain: out MVL7);
end Nchan;

architecture behavioral of Nchan is

begin
process

variable temp_output: MVL7;

begin
    if MVL7toBIT(Gate) = '1' then temp_output := '0';
    else temp_output := 'Z';
    end if;
    Drain <= temp_output;
    wait on Gate;
end process;
end behavioral;
```

Subcell NCHAN3 VHDL Model

```
-------------------------------------------------------
-- Date:  14 Sep 91
-- Version:  1
--
-- Unix filename:  nchan3.vhd
--
-- Function:  This file is a behavioral description of an
-- N-channel transistor which can be connected
-- in series and used as a switch
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity nchan3 is
  port(Gate: in MVL7;
       Drain: out DotX;
       Source: in MVL7);
end nchan3;

architecture behavioral of Nchan3 is
begin
process
variable temp_output: MVL7;
  begin
    if MVL7toBIT(Gate) = '0' then temp_output := 'Z';
    elsif Source = '0' then  temp_output := '0';
    else temp_output := 'Z';
    end if;
    Drain <= temp_output;
    wait on Gate, Source;
end process;
end behavioral;
```

Subcell TGATE VHDL Model

```
-------------------------------------------------------
-- Date:  24 Feb 91
-- Version:  6
--
-- Unix filename:  tgate.vhd
--
-- Function:  This file is a behavioral description of a transmission
-- gate. The input/output pins are 1 and 2. The true control input
-- is pin3, and the complemented control input is 4.
-- The file contains both the entity and behavioral architecture of the
-- transmission gate.
--    NOTE: The tgate is modeled as a 1-way device
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity tgate is
  port(p1 : in MVL7;
       p2 : in MVL7;
       g : in MVL7;
       d  : inout DotX);
end tgate;

architecture behavioral of tgate is
    Signal  node1 : MVL7;

begin
process

begin
    if ( MVL7toBIT(g) = '0' and MVL7toBIT(p1) = '1' and
         MVL7toBIT(p2) = '0') then d <= '0';
      elsif (MVL7toBIT(g)='1' and MVL7toBIT(p1) = '1' and
         MVL7toBIT(p2) = '0') then d <= '1';
    end if;
    if (d = '1') then Node1 <= 'H';
    end if;
    if (d = '0') then Node1 <= 'L';
    end if;
    if ((d = 'X') or (d = 'W') or (d = 'Z')) then Node1 <= 'Z';
    end if;
    if ( MVL7toBIT(p1) = '0' or
         MVL7toBIT(p2) = '1') then d <= Node1;
    end if;

    wait on g , p1, p2, d;
end process;
end behavioral;
```

Subcell PCHPU VHDL Model

```
-------------------------------------------------------
-- Date:  14 Sep 91
-- Version:  3
--
-- Unix filename:  pchpu.vhd
--
-- Function:  This file is a behavioral description of a
-- P-channel transistor connected as a pull-up resistor.
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity pchpu is
  port( PULLUP  : inout DotX);
end pchpu;

architecture behavioral of pchpu is
begin
process
variable temp_output: MVL7;
begin
    temp_output := 'H';
    PULLUP <= temp_output;
wait on PULLUP;
end process;
end behavioral;
```

Subcell TPAD VHDL Model

```
-------------------------------------------------------
-- Date:  14 Sep 91
-- Version:  2
--
-- Unix filename:  tpad.vhd
--
-- Function:  Dummy of a test pad.
-------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity TPAD is
    port(PIN1 : in MVL7);
end TPAD;

architecture behavioral of TPAD is
begin
process
begin
--dummy
wait;
end process;
end behavioral;
```

Appendix C: SCHEMA Pinlist to VHDL Structural Model Conversion Program

```
                                    ' SCHEMA Pin to vhdl net pin list conversion program
    VERNO$ = "V 0.98 (Developmental) - 8/11/91"
             ' Uses the pin list as input and creates a vhdl
             ' compatible definition as output.
             ' These routines use the fact
             ' that the pins are sorted in the .NPL file.
    PRINT "VHDL Generator "; VERNO$
    ON ERROR GOTO BATERROR
    OPEN "VHDBAT" FOR INPUT AS #5
    ON ERROR GOTO 0
    WHILE NOT EOF(5)
    INPUT #5, FILE$
    GOSUB MAKEVHDL
    WEND
    CLOSE #4
    SYSTEM

  '  exceptions
BATERROR:
    IF ERR <> 53 THEN GOTO FATLERR  'if file not found, ask for input
    RESUME NOBATCH
FATLERR:
    ON ERROR GOTO 0              'fatal error
    SYSTEM   'should never execute

NOBATCH:
    ON ERROR GOTO 0
    INPUT "Name of file to convert to VHDL : (no extension) "; FILE$
    GOSUB MAKEVHDL
ANOTHER:
    INPUT "Process another .NPL file (y/n)"; ANS$
    IF LEFT$(ANS$, 1) = "Y" THEN GOTO NOBATCH
    IF LEFT$(ANS$, 1) = "y" THEN GOTO NOBATCH
    IF LEFT$(ANS$, 1) = "N" THEN SYSTEM
    IF LEFT$(ANS$, 1) = "n" THEN SYSTEM
    PRINT "Improper input. Enter Y or y to process another file or"
    PRINT "                      N or n to exit program."
    GOTO ANOTHER
   ' end of top level (main) program

 'begin procedure
MAKEVHDL:
    PRINT "PROCESSING FILE "; FILE$; ".NPL"
    OPEN FILE$ + ".npl" FOR INPUT AS #1
    FERR = 0
    DIM PIN$(60), SIG$(60), ATTR$(60)
    DIM CMPS$(30), NSIG$(200), OSIG$(100)

    '        CMP$() IS COMPONENT NAMES, SIG$() IS NEW SIGNALS

    OX = 1
    NX = 1
    CX = 1
    OPEN FILE$ + ".VHD" FOR OUTPUT AS #2
    OPEN FILE$ + ".CMP" FOR OUTPUT AS #3
    PRINT #2, "-- VHDL GENERATED BY AVG FROM FILE "; FILE$; "(.NPL)"
    PRINT #2, "--       Automatic VHDL Generator (AVG) "; VERNO$
    PRINT #2, "--       Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from"
    PRINT #2, "--       the Omation SCHEMA SPICE netlist generator."
    PRINT #2, "--"
    PRINT #2, "--      Generated on "; DATE$; " at "; TIME$
    PRINT #2, "--"
    PRINT #2, "Library ZYCAD;"
    PRINT #2, "use ZYCAD.types.all;"
    PRINT #2, "use WORK.all;"
    PRINT #2, ""
    PRINT #2, "entity "; FILE$; " is"
```

```
    PRINT #3, "-- COMPONENT GENERATED BY AVG FROM FILE "; FILE$; "(.NPL)"
    PRINT #3, "--       Automatic VHDL Generator (AVG) "; VERNO$
    PRINT #3, "--             Generated on "; DATE$; " at "; TIME$
    PRINT #3, "--"
    PRINT #3, "component "; FILE$;
    INPUT #1, BRACKET$

    '    ***************** PROCESS SYMBOLS DESCRIPTIONS **************

    WHILE BRACKET$ <> "("
      INPUT #1, NNAME$, RNUM$, REFD$, PAGEN$
      LINE INPUT #1, DESC1$
      LINE INPUT #1, DESC2$
      LINE INPUT #1, DESC3$
      LINE INPUT #1, DESC4$
      INPUT #1, P1$, P2$, P3$, P4$
      INPUT #1, PIN$(1)

      '    *********** PLACE THE PINS INTO AN ARRAY **********


      FOR I = 1 TO 1000
        INPUT #1, ATTR$(I), SIG$(I), SIGNUM$
        IF SIG$(I) = "+5" THEN SIG$(I) = "VDD"
        IF LEFT$(SIG$(I), 1) <> "$" GOTO 520
        SIG$(I) = MID$(STR$(1000 + VAL(SIGNUM$)), 2)
520     INPUT #1, PIN$(I + 1)
        IF PIN$(I + 1) = "]" GOTO 550
      NEXT I

550   INPUT #1, BRACKET$

      '   ******** PROCESS COMPONENT TYPE BY REFD FIRST LETTER *******

      TYPE$ = LEFT$(REFD$, 1)
      IF TYPE$ <> "Z" GOTO 960
      PRINT #2, "     port("
      PRINT #3, "     port("

      FOR L = 1 TO (I - 1)
         OSIG$(L) = MID$(SIG$(L), 2)
         IF LEFT$(OSIG$(L), 5) <> "DOTX_" THEN GOTO 710
         PRTYPE$ = "inout DotX"
         OSIG$(L) = MID$(OSIG$(L), 6)
         GOTO 760
710      IF LEFT$(ATTR$(L), 1) = "I" THEN PRTYPE$ = "in MVL7"
         IF LEFT$(ATTR$(L), 1) = "B" THEN PRTYPE$ = "inout MVL7"
         IF LEFT$(ATTR$(L), 1) = "T" THEN PRTYPE$ = "inout MVL7"
         IF LEFT$(ATTR$(L), 1) = "X" THEN PRTYPE$ = "inout MVL7"
         IF LEFT$(ATTR$(L), 1) = "O" THEN PRTYPE$ = "out MVL7"
760      PRINT #2, OSIG$(L); ": "; PRTYPE$; ";"
         PRINT #3, OSIG$(L); ": "; PRTYPE$; ";"
      NEXT L

      OSIG$(L) = MID$(SIG$(L), 2)
      IF LEFT$(OSIG$(L), 5) <> "DOTX_" THEN GOTO 840
      PRTYPE$ = "inout DotX"
      OSIG$(L) = MID$(OSIG$(L), 6)
      GOTO 890
840   IF LEFT$(ATTR$(L), 1) = "I" THEN PRTYPE$ = "in MVL7"
      IF LEFT$(ATTR$(L), 1) = "B" THEN PRTYPE$ = "inout MVL7"
      IF LEFT$(ATTR$(L), 1) = "T" THEN PRTYPE$ = "inout MVL7"
      IF LEFT$(ATTR$(L), 1) = "X" THEN PRTYPE$ = "inout MVL7"
      IF LEFT$(ATTR$(L), 1) = "O" THEN PRTYPE$ = "out MVL7"
890
      OX = L
      PRINT #2, OSIG$(L); ": "; PRTYPE$; " );"
```

```
      PRINT #3, OSIG$(L); ": "; PRTYPE$; ");"
      GOTO 980
960   GOSUB 1830
      GOSUB 2120
980 WEND
    '        -- END THE WHILE LOOP. READ UNTIL OPEN PAREN
    PRINT #2, "end "; FILE$; ";"
    PRINT #3, "end component;"
    PRINT #2, "-- END OF AUTOMATICALLY GENERATED VHDL entity"
    PRINT #2, ""
    PRINT #3, ""
    CLOSE #3
    PRINT #2, "architecture structure of "; FILE$; " is"
    CLOSE #1
    PRINT #2, ""
    PRINT #2, "-- SIGNAL DECLARATIONS HERE"
    GOSUB 1910
    PRINT #2, "-- COMPONENT DECLARATIONS HERE"
    PRINT #2, ""
    GOSUB 2240
    PRINT #2, ""
    PRINT #2, "begin"
    PRINT #2, ""

    '  Second pass through input file to generate structure.

    OPEN FILE$ + ".npl" FOR INPUT AS #1
    PRINT #2, ""
    INPUT #1, BRACKET$

    ' ***************** PROCESS SYMBOLS DESCRIPTIONS **************

    WHILE BRACKET$ <> "("
       INPUT #1, NNAME$, RNUM$, REFD$, PAGEN$
       LINE INPUT #1, DESC1$
       LINE INPUT #1, DESC2$
       LINE INPUT #1, DESC3$
       LINE INPUT #1, DESC4$
       INPUT #1, P1$, P2$, P3$, P4$
       INPUT #1, PIN$(1)
       IF PIN$(1) = "+5" THEN PIN$(1) = "VDD"

       ' *********** PLACE THE PINS INTO AN ARRAY **********

       FOR I = 1 TO 1000
          INPUT #1, ATTR$, SIG$(I), SIGNUM$
          IF SIG$(I) = "+5" THEN SIG$(I) = "VDD"
          IF LEFT$(SIG$(I), 5) = "DOTX_" THEN SIG$(I) = MID$(SIG$(I), 6)
          IF LEFT$(SIG$(I), 1) <> "$" GOTO 1400
          SIG$(I) = MID$(STR$(1000 + VAL(SIGNUM$)), 2)
1400      INPUT #1, PIN$(I + 1)
          IF PIN$(I + 1) = "]" GOTO 1430
       NEXT I
1430   INPUT #1, BRACKET$

       ' ******** PROCESS COMPONENT TYPE BY REFD FIRST LETTER *******

       TYPE$ = LEFT$(REFD$, 1)
       IF TYPE$ = "Z" GOTO 1680
       PRINT #2, REFD$; ": "; NNAME$; " PORT MAP ( ";
       FOR L = 1 TO (I - 1)
           S$ = SIG$(L)
           IF S$ = "" THEN S$ = "OPEN"
           IF (LEFT$(S$, 1) >= "0" AND LEFT$(S$, 1) <= "9") THEN S$ = "S" + S$
           IF ((L MOD 5) = 0) GOTO 1580 ELSE GOTO 1560
1560       PRINT #2, S$; ", ";
           GOTO 1600
```

```
1580        PRINT #2, S$; ","
            PRINT #2, "                          ";
1600    NEXT L
        S$ = SIG$(L)
        IF S$ = "" THEN S$ = "OPEN"
        IF (LEFT$(S$, 1) >= "0" AND LEFT$(S$, 1) <= "9") THEN
        S$ = "S" + S$
        END IF
1650    PRINT #2, S$; ");"
        PRINT #2, ""
        GOTO 1690
1680    ' SKIP Z HERE (SYMBOL DEF)
1690 WEND
     '      -- END THE WHILE LOOP. READ UNTIL OPEN PAREN
     PRINT #2, "-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE"
     PRINT #2, ""
     PRINT #2, "end structure ;"
     CLOSE #1, #2, #3, #4
     ERASE PIN$, SIG$, ATTR$
     ERASE CMPS$, NSIG$, OSIG$
     IF FERR = 1 THEN PRINT "Output file has error(s). Will not analyze properly."
     RETURN
     '     END PROCEDURE


   ' begin procedure
1830 FOR L = 1 TO I
        FOR W = 1 TO NX
            IF NSIG$(W) = SIG$(L) THEN GOTO LBLTST
        NEXT W
        NSIG$(NX) = SIG$(L)
        NX = NX + 1
LBLTST:
     NEXT L
     RETURN
   'end procedure

    'begin procedure
1910
           '  WRITE SIGNAL DECLARATIONS TO VHDL FILE
     PRINT #2, "         signal GND : MVL7 :='0';"
     PRINT #2, "         signal VDD : MVL7 :='1';"
     FOR L = 1 TO NX - 1
         S$ = NSIG$(L)
         IF S$ = "VDD" THEN GOTO 2070
         IF S$ = "GND" THEN GOTO 2070
         IF S$ = "" THEN GOTO 2070
         DXFLAG = 0
         IF LEFT$(S$, 5) <> "DOTX_" THEN GOTO 2000
         DXFLAG = 1
         S$ = MID$(S$, 6)
2000     FOR W = 1 TO OX
             IF OSIG$(W) = S$ THEN GOTO 2070
         NEXT W
         IF LEFT$(S$, 1) > "9" THEN GOTO 2060
         IF LEFT$(S$, 1) < "0" THEN GOTO 2060
         S$ = "S" + S$
2060     IF DXFLAG = 0 THEN PRINT #2, "    signal  "; S$; " : MVL7;"
         IF DXFLAG = 1 THEN PRINT #2, "    signal  "; S$; " : DotX;"
2070 NEXT L
     PRINT #2, ""
     PRINT #2, "--       END SIGNALS "
     PRINT #2, ""
     RETURN
    'end procedure
```

```
    'begin procedure
2120
     '                    SAVE COMPONENTS IN CMP$ ARRAY
     IF CX > 1 THEN GOTO 2170
     CMP$(CX) = NNAME$
     CX = CX + 1
     GOTO 2230
2170 FOR W = 1 TO CX - 1
         IF CMP$(W) = NNAME$ THEN GOTO 2220
     NEXT W
     CMP$(CX) = NNAME$
     CX = CX + 1
2220 ' ALREADY STORED
2230 RETURN
    'end procedure



2240
       '  READ IN COMPONENT DECLARATIONS FROM EXTERNAL FILES
     FOR R = 1 TO CX - 1
         ON ERROR GOTO 2360
         OPEN CMP$(R) + ".CMP" FOR INPUT AS #4
         WHILE NOT EOF(4)
             LINE INPUT #4, LX$
             PRINT #2, LX$
         WEND
         CLOSE #4
2330     ON ERROR GOTO 0
     NEXT R
     RETURN
   '         exceptions
2360
     IF ERR <> 53 THEN GOTO 2410
     PRINT #2, "-- FILE "; CMP$(R) + ".CMP"; "  NOT AVAILABLE. "
     PRINT #2, "-- PUT COMPONENT DEFINITION FOR "; CMP$(R); " HERE."
     PRINT #2, ""
     PRINT " ERROR - COMPONENT DESCRIPTION "; CMP$(R) + ".CMP"; " NOT FOUND"
     PRINT " Output File is not useable without this description. "
     PRINT ""
     FERR = 1
     RESUME 2330
2410 ON ERROR GOTO 0
  'end procedure
```

Appendix D: Microcircuit 1 Test Fixture Schematic Diagram.

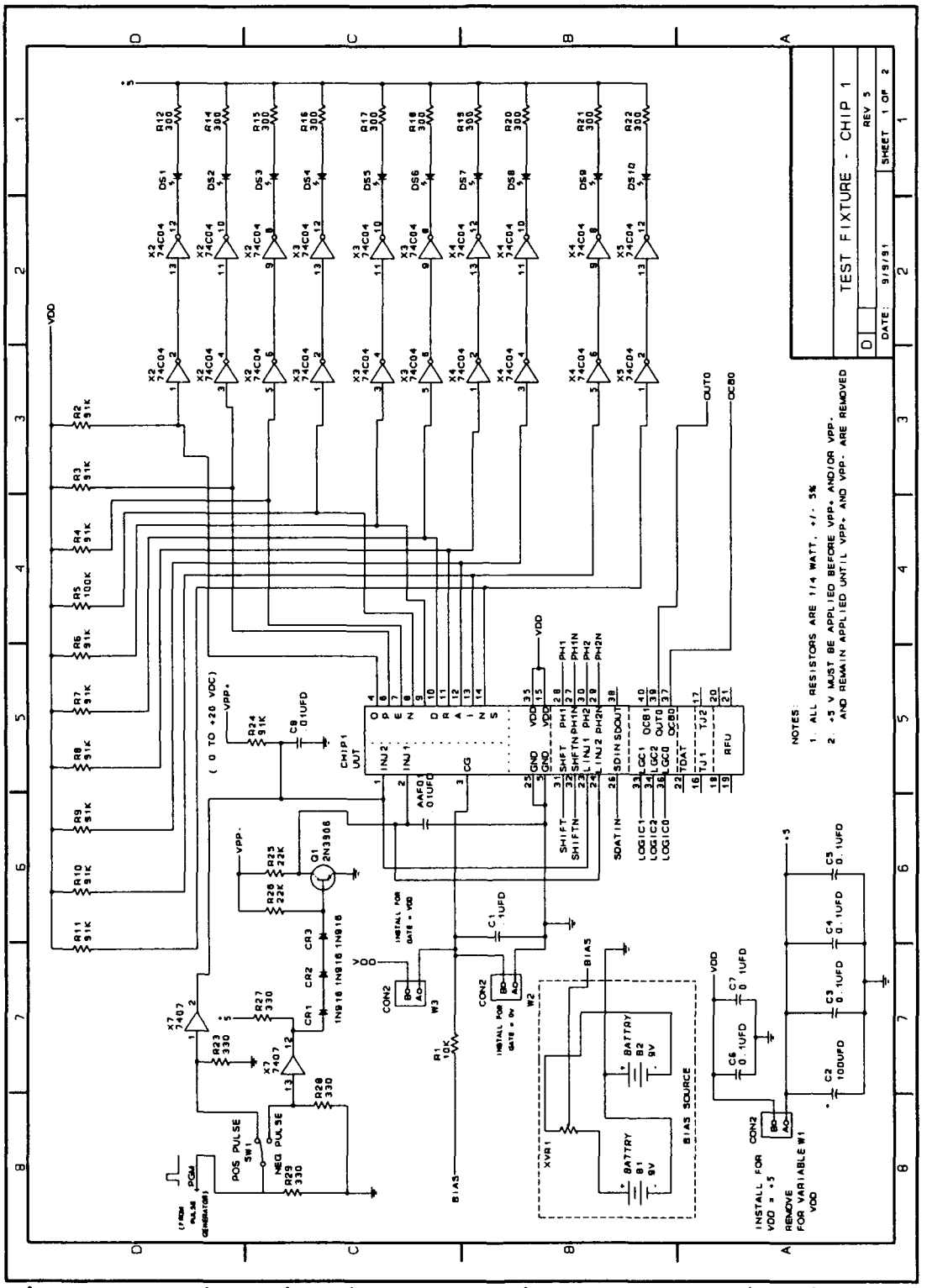


Figure 19. Microcircuit 1 Test Fixture Schematic Diagram. Sheet 1 of 2.

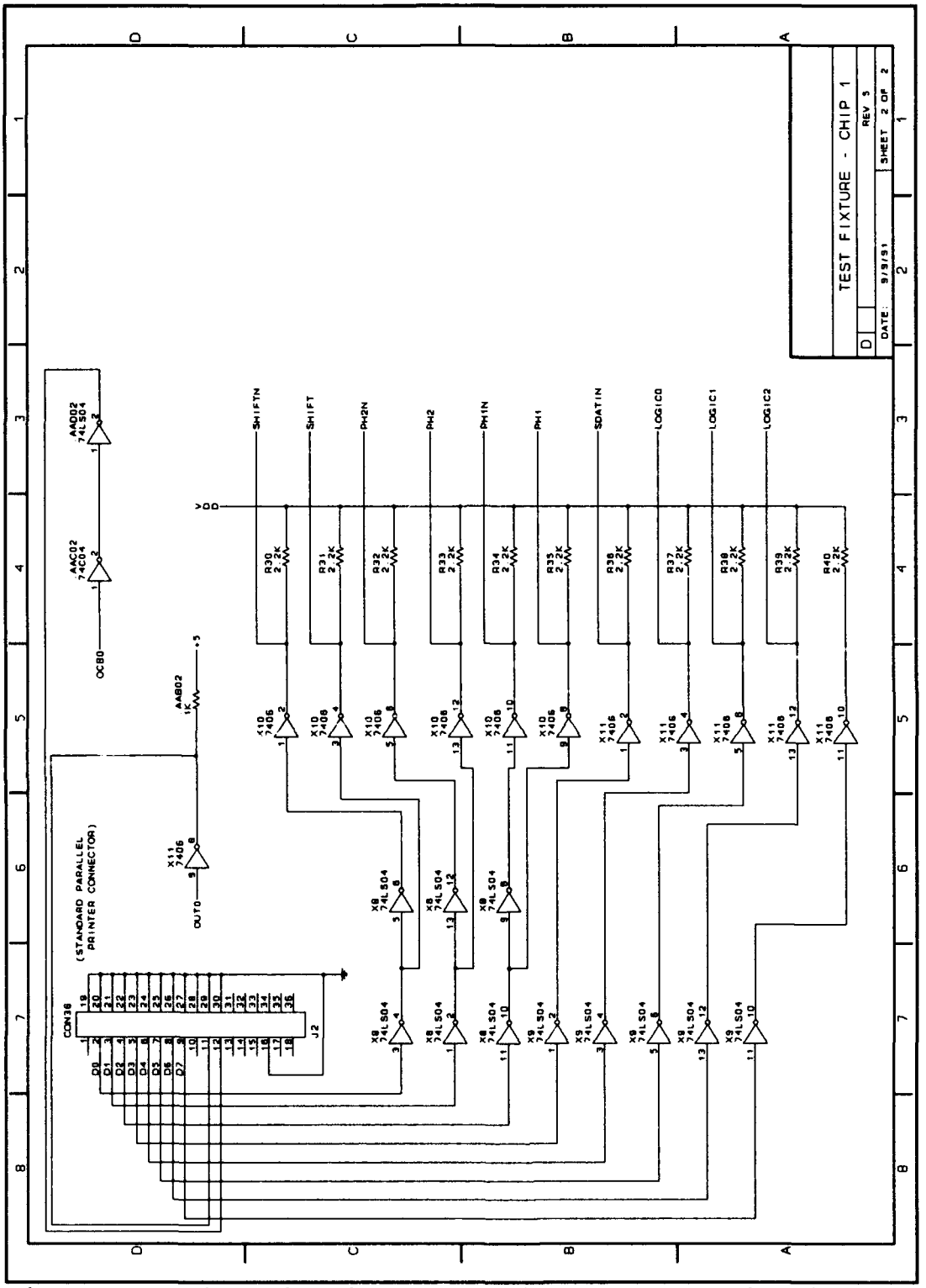


Figure 20. Microcircuit 1 Test Fixture Schematic Diagram. Sheet 2 of 2.

Appendix E: Microcircuit 1 Test Fixture Control Program.

```
program eproj;
{  Program Name:          EPROJ
   Written By:            Joseph V. Breen
   Date Last Modified:    September 25, 1991
   Purpose:               Provides a menu driven interface which
   controls  the test fixture for the N15S MC-1
   programmable logic prototype microcircuit.
   Functions to load the shift register,  assist in erasing,
   and allow programmable logic function verification are provided.
}
const
{ Masks For Data Bits Of Printer Port Byte }
  shift: byte  = 1;
  ph2: byte = 2;
  ph1: byte = 4;
  sdat: byte = 8;
  logic0: byte = 16;
  logic1: byte = 32;
  logic2:byte = 64;
  addrs: integer = $378;
  stadr: integer = $379;
VAR
  itemp: integer;
  i : integer;
  tlog0, tlog1, tlog2 : integer;
  temp: BYTE;
  temp2 : byte;
  ch: char;
  istrg: string[20];
  pw2,pw1,pw0 : BYTE;

procedure putdat(databyte : byte);
begin
   port[addrs] := databyte;
end;

procedure ordat(odata : byte);
VAR
   obyte : byte;
begin
   obyte := port[addrs];
   obyte := obyte or odata;
   port[addrs] := obyte;
end;

procedure anddat(odata : byte);
VAR
   obyte : byte;
begin
   obyte := port[addrs];
   obyte := obyte and not odata;
   port[addrs] := obyte;
end;

Procedure toggl(odata : byte);
begin
     ordat(odata);
     anddat(odata);
end;

FUNCTION HCHAR(inchr:INTEGER) : CHAR;
         begin
         CASE INCHR OF
           0 : HCHAR := '0';
           1 : HCHAR := '1';
           2 : HCHAR := '2';
           3 : HCHAR := '3';
```

```
          4 : HCHAR := '4';
          5 : HCHAR := '5';
          6 : HCHAR := '6';
          7 : HCHAR := '7';
          8 : HCHAR := '8';
          9 : HCHAR := '9';
         10 : HCHAR := 'A';
         11 : HCHAR := 'B';
         12 : HCHAR := 'C';
         13 : HCHAR := 'D';
         14: HCHAR := 'E';
         15 : HCHAR := 'F';
         ELSE WRITELN('HCHAR ERROR !!!!',INCHR);
        end;
end;

procedure writehex(hexchars : byte);
var
tcharh : char;
begin
                tcharh :=HChAR(hexchars div 16);
                write(tcharh);
                tcharh :=HChAR(hexchars mod 16);
                write(tcharh);
end;

Procedure get_integer(var inumb: integer);
var
  icode : integer ;
  instg : string[20];
  tempint : integer;
begin
   icode := 1;
   writeln('');
   repeat
   write('enter number :');
   readln(instg);
   val(instg,tempint,icode);
   if(icode <> 0) then
     begin
       writeln('bad integer. Try again.');
     end;
   until icode = 0;
   write('integer ',tempint,' entered');
   inumb := tempint;
end;

Procedure get_byte(var ibyte: byte);
var
  icode : integer ;
  instg : string[20];
  tempint : integer;
begin
   icode := 1;
   writeln('');
   repeat
   write('enter byte :');
   readln(instg);
   val(instg,tempint,icode);
   if(icode <> 0) then
     begin
       writeln('bad number. Try again.');
     end;
     if (tempint > 255) then
        begin
          writeln('Maximum exceeded. Maximum is 255 for a Byte value. Try again.');
          icode :=1;
```

```
        end;
      until icode = 0;
   ibyte:=tempint;
   write('Byte :  ',ibyte,' entered');
end;

Procedure clok;
begin
    toggl(ph1);
    toggl(ph2);
end;

Procedure ptable;
begin
    temp := port[addrs];
    temp := temp div 2;
    temp := temp div 2;
    temp := temp div 2;
    temp := temp div 2;
    Tlog0  := temp mod 2;
    temp := temp div 2;
    Tlog1  := temp mod 2;
    temp := temp div 2;
    Tlog2  := temp mod 2;
    temp := port[stadr];
    temp := temp div 128;
    itemp := temp;
writeln('       ',tlog2,'         ',tlog1,'        ',tlog0,'            ',itemp);
end;

Procedure qtable;
begin
    temp := port[addrs];
    temp := temp div 2;
    temp := temp div 2;
    temp := temp div 2;
    temp := temp div 2;
    Tlog0  := temp mod 2;
    temp := temp div 2;
    Tlog1  := temp mod 2;
    temp := temp div 2;
    Tlog2  := temp mod 2;
    temp := port[stadr];
    temp := temp div 32;
    temp := temp and 1;
    itemp := temp;
writeln('       ',tlog2,'         ',tlog1,'        ',tlog0,'            ',itemp);
end;

Procedure pmenu;
begin;
writeln('            Action Menu For N15S MC-1 Test Program.');
writeln('');
writeln('     A    Set Shift True');
writeln('     B    Set Shift False');
writeln('     C    Clock Multiple Times (Prompt For Number)');
writeln('     D    Default Signals (Set Shift True, others False)');
writeln('     E    Examine. Build Truth Table for Logic 0-2 to Out0.');
writeln('     F    Int. Examine. Build Truth Table for Logic 0-2 to Wired-Or (37) ');
writeln('     L    Load Byte (8 bits) into Serial Register (Prompt For Byte)');
writeln('     M    Load Lower n bits of Byte into Serial Register (Prompt for Byte & #) ');
writeln('     P    Print Current States of Each Signal');
writeln('     Q    Quit Program');
writeln('     R    Reset Input Data to 0');
writeln('     S    Set Input Data to 1');
writeln('     T    Single Tick of Phase 1 Then Phase 2 Clocks');
writeln('     U    Prepare for ERASE (PH1 & 2 true, Input = 1)');
```

```
writeln('     V    Prepare for read (PH1 & 2 True, Input = 0)');
writeln(' W    Write Program Word        X    Get and Store Prog. Word');
writeln(' Y    Set Ph1 & Ph2 False(Dflt) Z    Set Ph1 & Ph2 True');
writeln(' 0    Set Logic 0 Input True    3    Set Logic 0 Input False');
writeln(' 1    Set Logic 1 Input True    4    Set Logic 1 Input False');
writeln(' 2    Set Logic 2 Input True    5    Set Logic 2 Input False');
end;


  begin
    putdat(0);
WRITELN;
WRITELN('         TEST PROGRAM FOR N15S MC-1         ');
WRITELN('       SPECIAL EEPROM TEST MICROCIRCUIT     ');
WRITELN('       September 18, 1991         V1.5      ');
WRITELN('       Written By:   Joe Breen, AFIT EN     ');
WRITELN;

begin
{Make Shift true, both clocks  and data false}
   putdat(0);
   ordat(shift);
   pmenu;

   repeat
      read(kbd,ch);
      ch := upcase(ch);
      write(ch);
      case ch of {main action loop}
        'A' : begin
                ordat(shift);
              end;

        'B' : begin
                anddat(shift);
              end;

        'C' : begin
                get_integer(itemp);
                writeln(' Clocking ',itemp, '  times.');
                for I := 1 to itemp do clok;
                writeln('Done.');
              end;

        'D' : begin
                putdat(shift);    {    only shift is true  }
              end;

       'E' : begin
              Writeln('             Truth Table');
              Writeln('      Logic2     Logic1     Logic0  :::   Out0');
              anddat(logic0);
              anddat(logic1);
              anddat(logic2),
              ptable;
              ordat(logic0);
              ptable;
              anddat(logic0);
              ordat(logic1);
              ptable;
              ordat(logic0);
              ptable;
              anddat(logic0);
              anddat(logic1);
              ordat(logic2);
              ptable;
              ordat(logic0);
```

```
        ptable;
        anddat(logic0);
        ordat(logic1);
        ptable;
        ordat(logic0);
        ptable;
        writeln('');
       end;

'F' : begin
        Writeln(' Wired OR (Pin 37)    Truth Table');
        Writeln('      Logic2     Logic1     Logic0  :::   WiredOr Out');
        anddat(logic0);
        anddat(logic1);
        anddat(logic2);
        qtable;
        ordat(logic0);
        qtable;
        anddat(logic0);
        ordat(logic1);
        qtable;
        ordat(logic0);
        qtable;
        anddat(logic0);
        anddat(logic1);
        ordat(logic2);
        qtable;
        ordat(logic0);
        qtable;
        anddat(logic0);
        ordat(logic1);
        qtable;
        ordat(logic0);
        qtable;
        writeln('');
      end;

  'L' : begin
          get_byte(temp);
          writeln(' Byte Value  ',temp, ' accepted.');
          for I := 1 to 8  do
          begin
             if ((temp mod 2) = 1) then anddat(sdat)
             else ordat(sdat);
             clok;
             writeln(temp mod 2);
             temp := temp div 2;
          end;
          writeln('Done.');
        end;

  'M' : begin
          writeln('');
          writeln('How many bits of data do you want to load (0 - 8)?');
          get_byte(temp);
          writeln('');
          if temp <> 0 then
          begin
          temp := temp - 1;
          temp := temp mod 8;
          temp2 := temp;
          writeln('What is the data byte to be used?');
          get_byte(temp);
          writeln(' Byte Value  ',temp, ' accepted.');
          for I := 0 to temp2  do
          begin
             if ((temp mod 2) = 1) then anddat(sdat)
```

```
            else ordat(sdat);
            clok;
            writeln(temp mod 2);
            temp := temp div 2;
         end;
         end;
         writeln('Done.');
      end;

'P' : begin
        temp := port[addrs];
         writeln(' Shift  =  ', temp mod 2);
         temp := temp div 2;
         writeln(' Phase 2  =  ', temp mod 2);
         temp := temp div 2;
         writeln(' Phase 1  =  ', temp mod 2);
         temp := temp div 2;
         writeln(' SDAT (Serial Data Input)   =  ', temp mod 2);
         temp := temp div 2;
         writeln(' Logic0  =  ', temp mod 2);
         temp := temp div 2;
         writeln(' Logic1  =  ', temp mod 2);
         temp := temp div 2;
         writeln(' Logic2  =  ', temp mod 2);
         temp := temp div 2;
      end;

'Q' :;  { do nothing - going to quit}

'R' : begin
         anddat(sdat);
      end;

'S' : begin
         ordat(sdat);
      end;

'T' : begin
         clok;
      end;

'U' : begin
         ordat(ph1);
         ordat(ph2);
         ordat(sdat);
      end;

'V' : begin
         ordat(ph1);
         ordat(ph2);
         anddat(sdat);
      end;

'W' : begin
         clok;
         temp := pw0;
         for I := 1 to 8  do
         begin
            if ((temp mod 2) = 1) then anddat(sdat)
            else ordat(sdat);
            clok;
            temp := temp div 2;
         end;
         temp := pw1;
         for I := 1 to 8  do
         begin
            if ((temp mod 2) = 1) then anddat(sdat)
```

```
            else ordat(sdat);
            clok;
            temp := temp div 2;
          end;
          temp := pw2;
          for I := 1 to 6  do
          begin
            if ((temp mod 2) = 1) then anddat(sdat)
            else ordat(sdat);
            clok;
            temp := temp div 2;
          end;
        writeln('');
        write ('Program Word  ');
        writehex(pw2);
        writehex(pw1);
        writehex(pw0);
        writeln('  (hex) is written');
 end;

'X' : begin
        writeln(' Use $ for Hexadecimal entry  ');
        writeln(' Enter Most Sig. Byte of Program Word.');
        get_byte(pw2);
        writeln(' Enter Middle Byte of Program Word.');
        get_byte(pw1);
        writeln(' Enter Least Sig. Byte of Program Word.');
        get_byte(pw0);
        writeln('');
        write ('Program Word  ');
        writehex(pw2);
        writehex(pw1);
        writehex(pw0);
        writeln('  (hex) is entered');
      end;

'Y' : begin
        anddat(ph1);
        anddat(ph2);
      end;

'Z' : begin
        ordat(ph1);
        ordat(ph2);
      end;

'0' : begin
        ordat(logic0);
      end;

'3' : begin
        anddat(logic0);
      end;

'1' : begin
        ordat(logic1);
      end;

'4' : begin
        anddat(logic1);
      end;

'2' : begin
        ordat(logic2);
      end;

'5' : begin
```

```
                anddat(logic2);
            end;

        else begin
            sound(510);
            delay(300);
            nosound;
            writeln('');
            writeln(' Unrecognized input ',ch);
            pmenu;
          end;
    end;

    until ((ch = 'q') or (ch = 'Q'));
  end;
end.
```

Appendix F: Microcircuit 2

Schematic Diagrams.

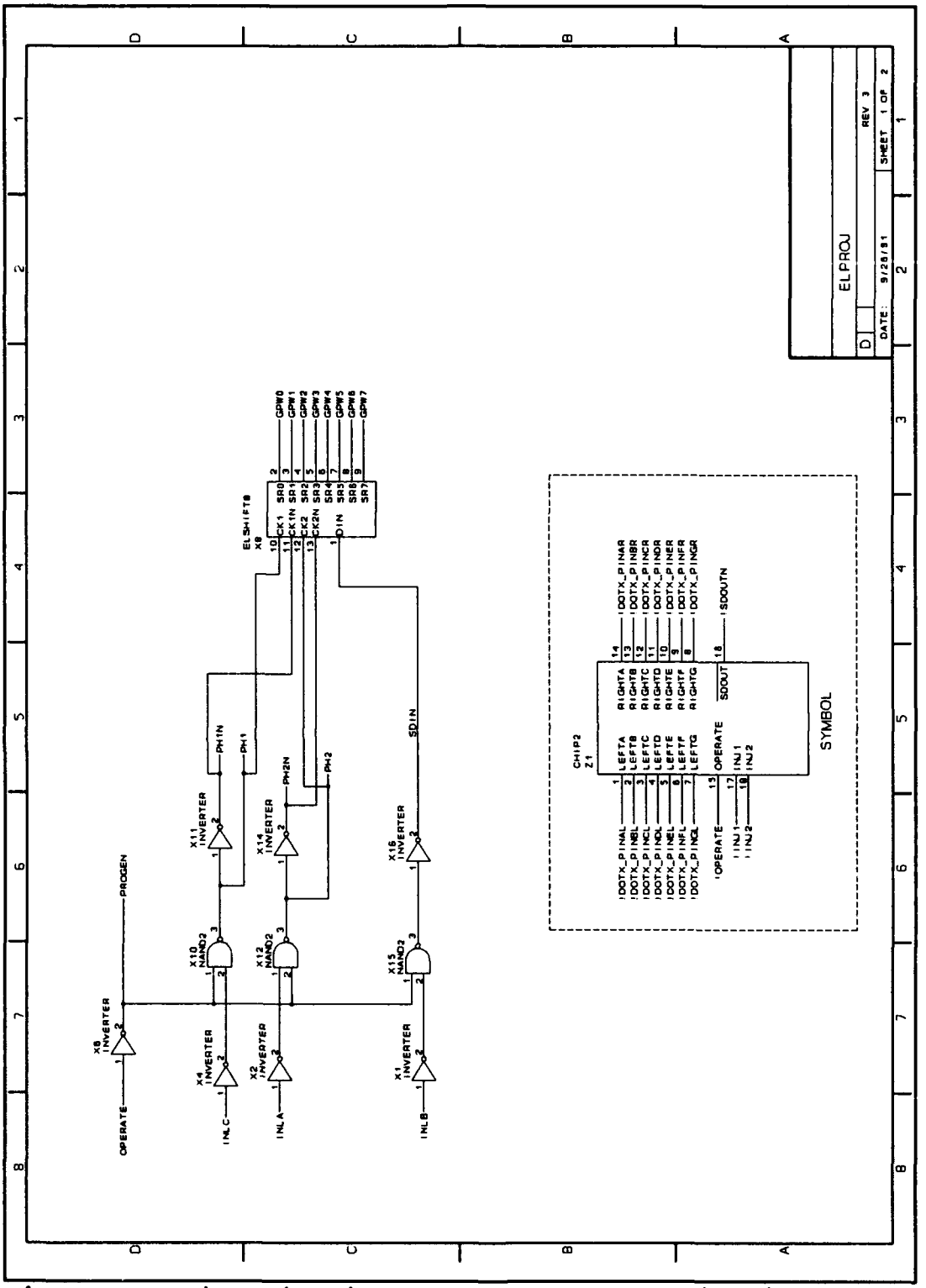


Figure 21. Microcircuit 2 Top Level Schematic Diagram. Sheet 1 of 2.

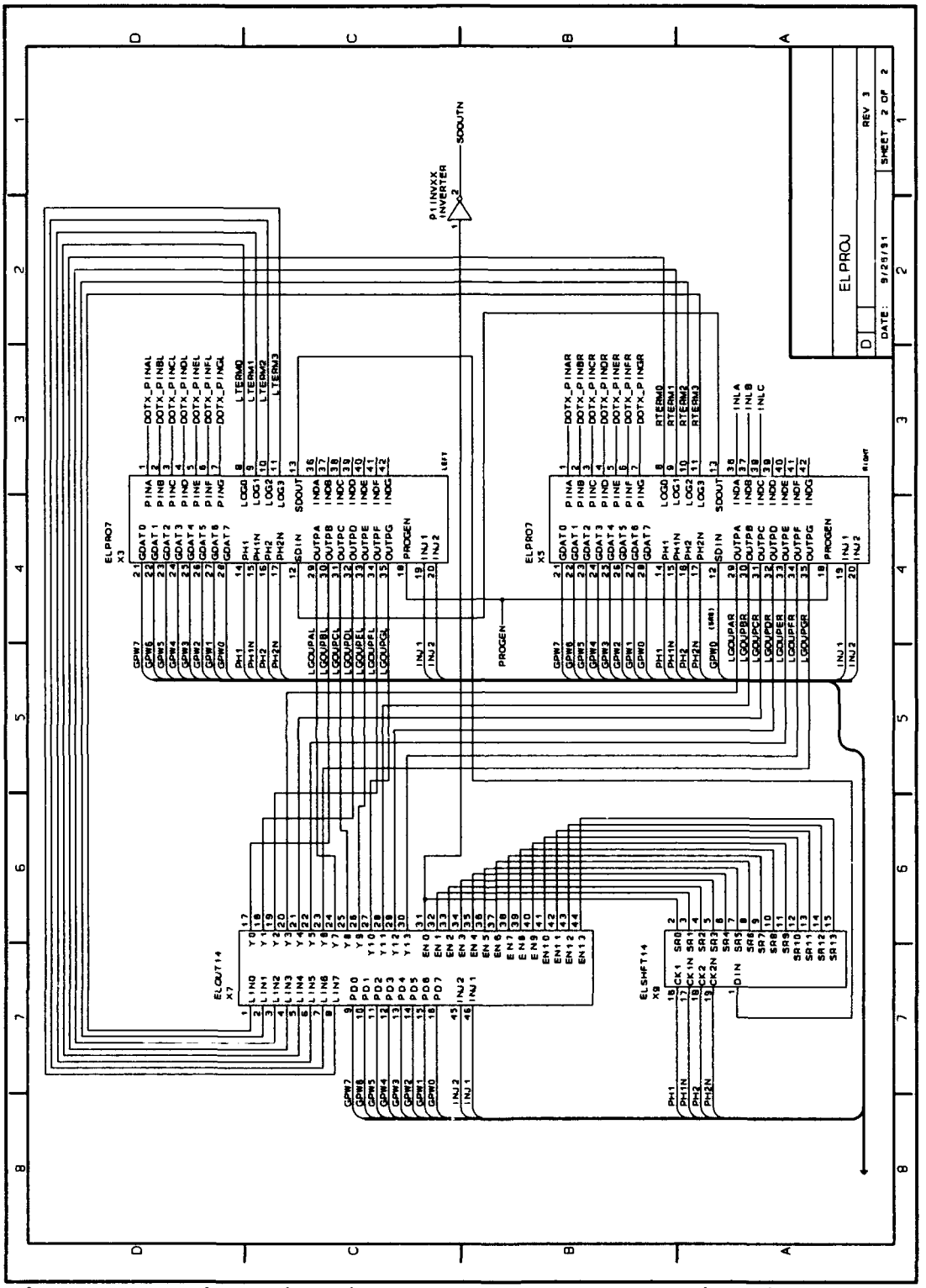


Figure 22. Microcircuit 2 Top Level Schematic Diagram. Sheet 2 of 2.

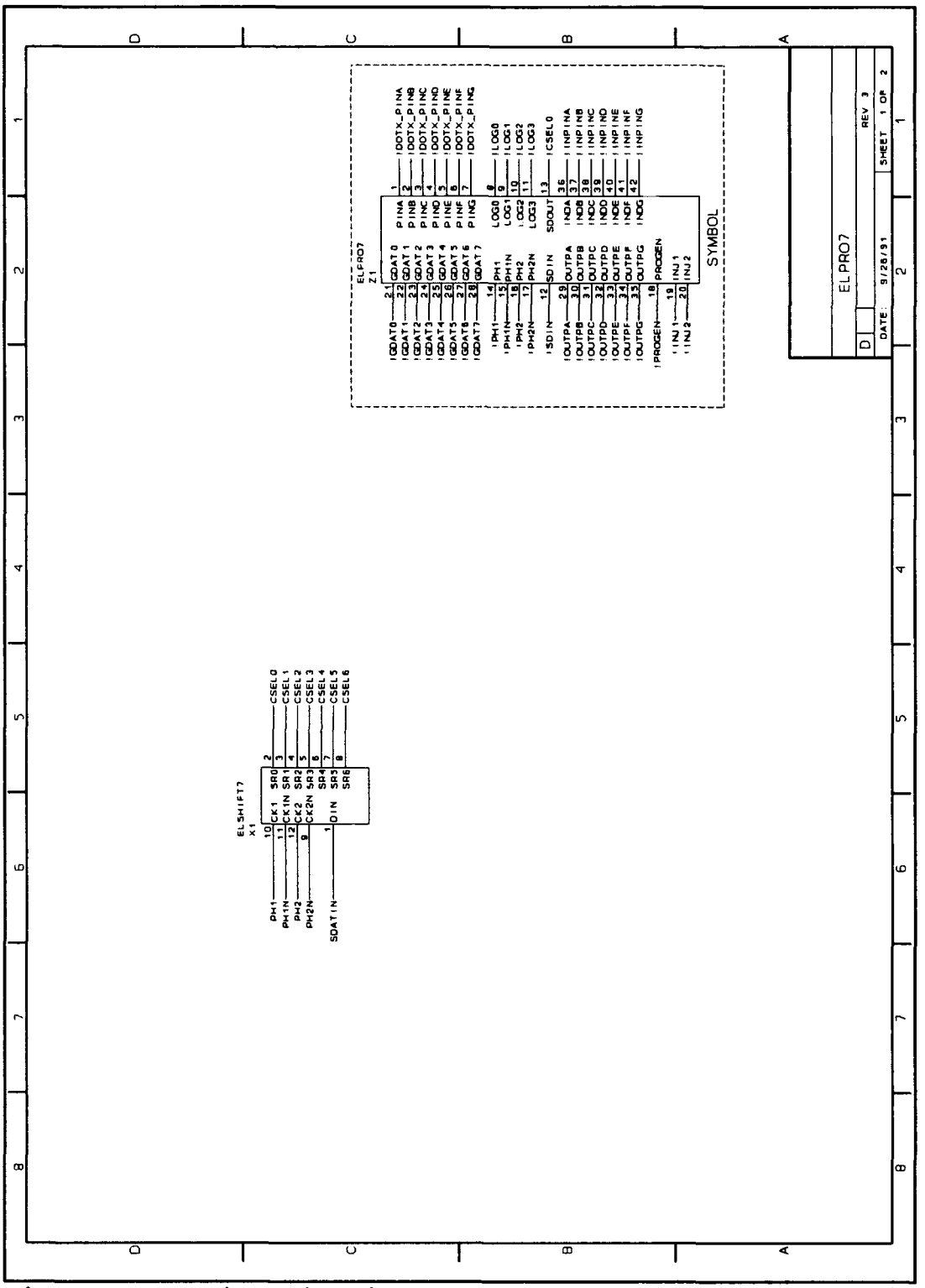


Figure 23. Microcircuit 2 Subcell ELPRO7 Schematic Diagram. Sheet 1 of 2.

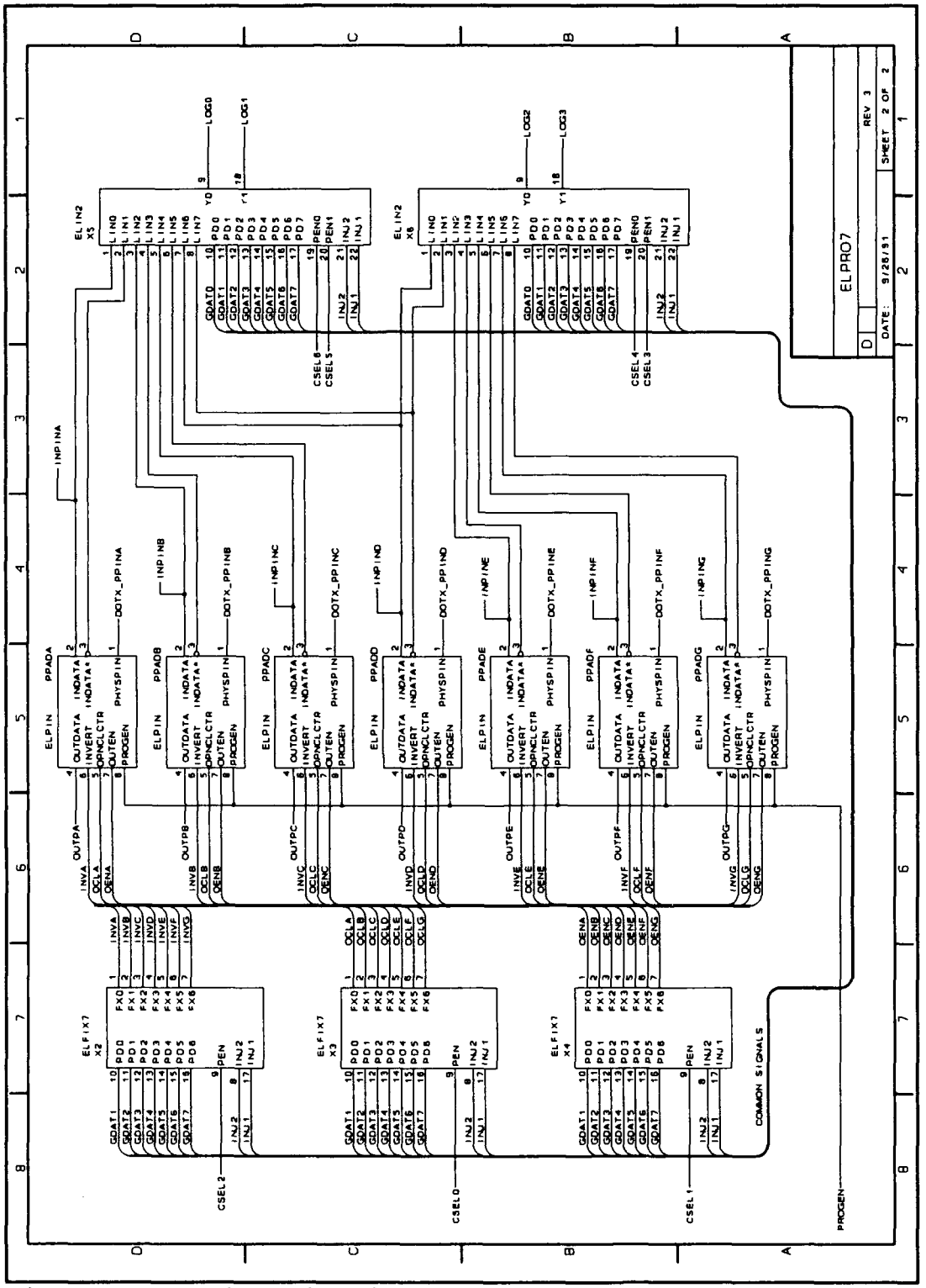


Figure 24. Microcircuit 2 Subcell ELPRO7 Schematic Diagram. Sheet 2 of 2.

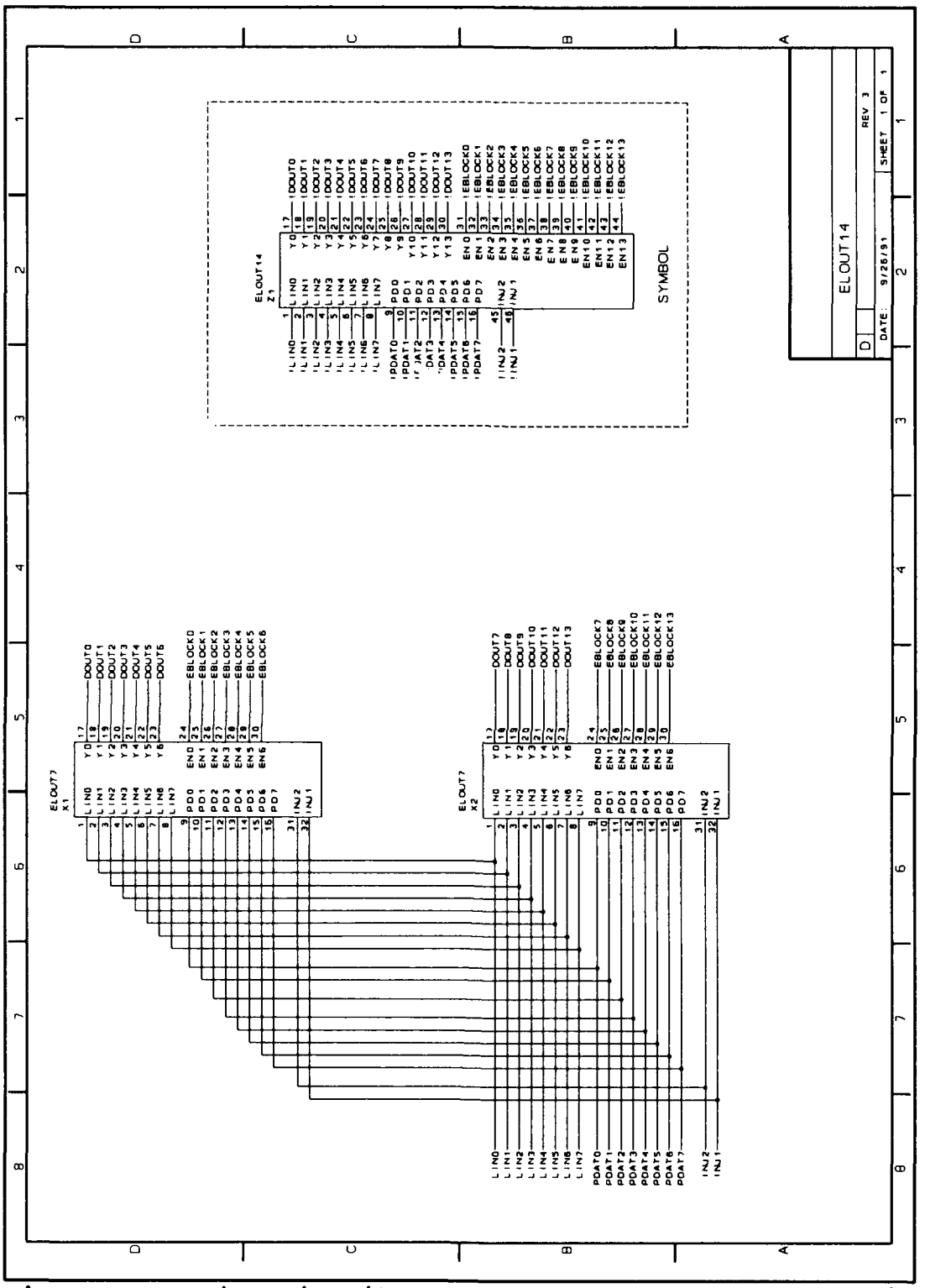


Figure 25. Microcircuit 2 Subcell ELOUT14 Schematic Diagram

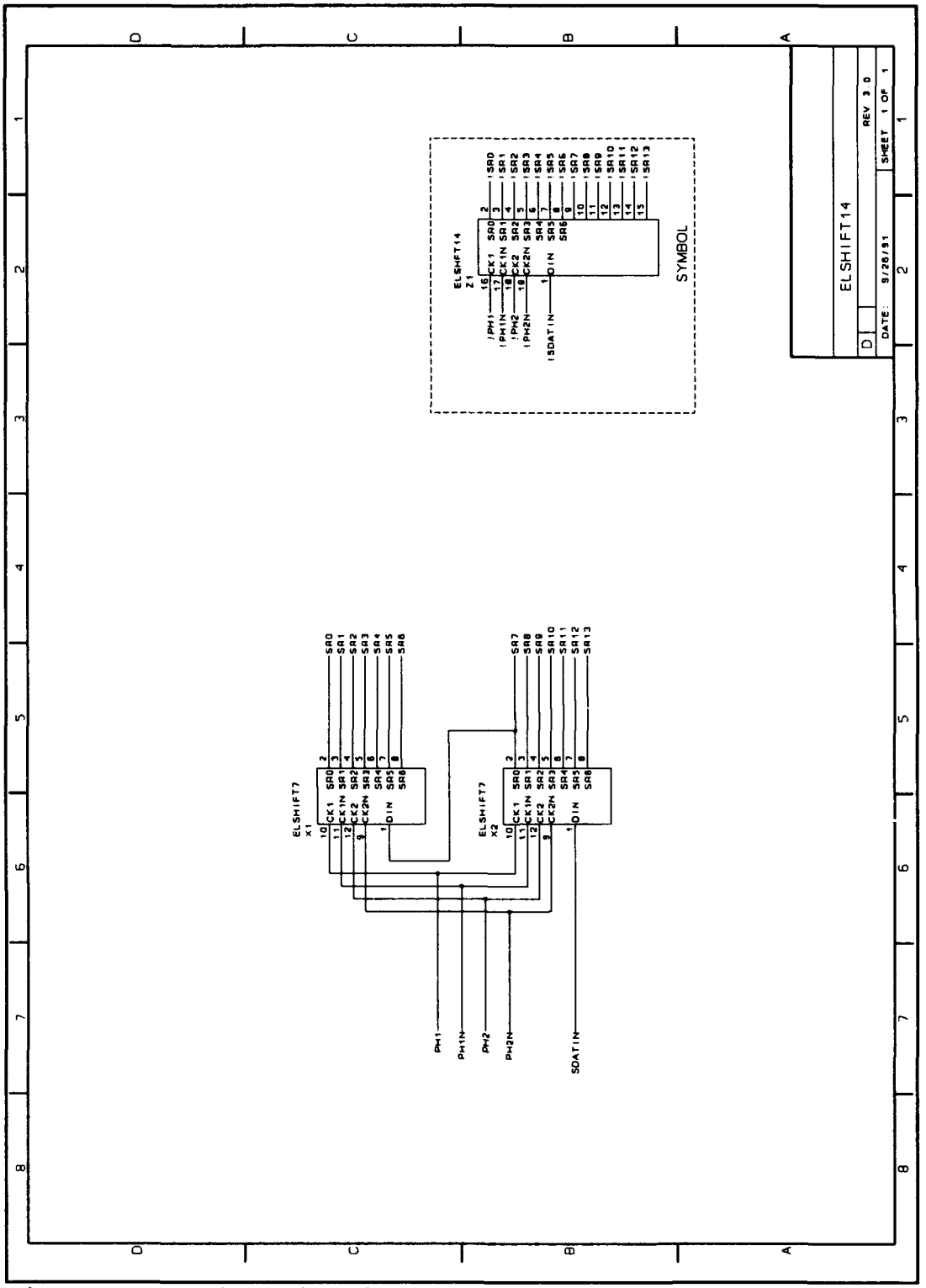


Figure 26. Microcircuit 2 Subcell ELSHFT14 Schematic Diagram.

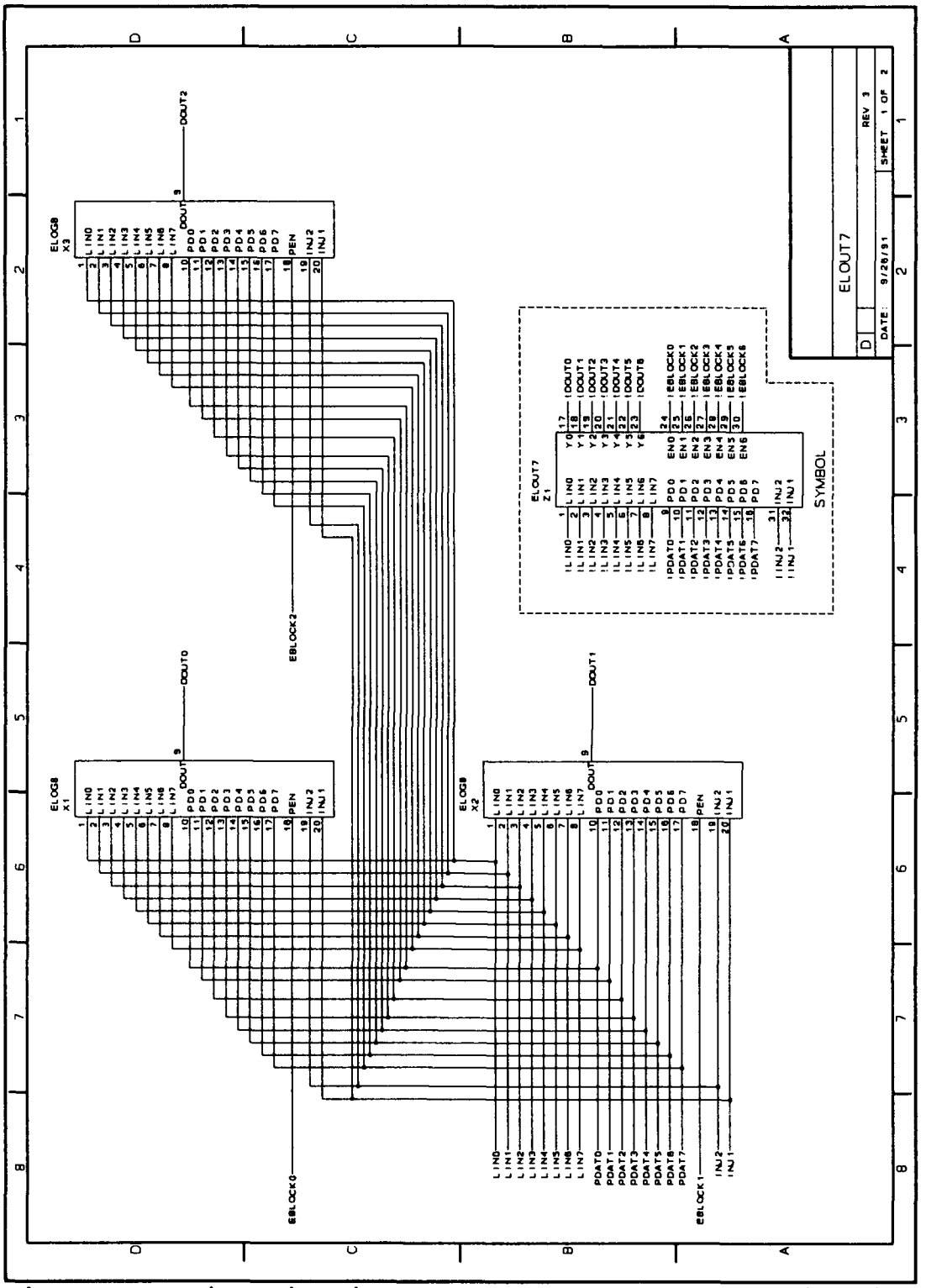


Figure 27. Microcircuit 2 Subcell ELOUT7 Schematic Diagram. Sheet 1 of 2.

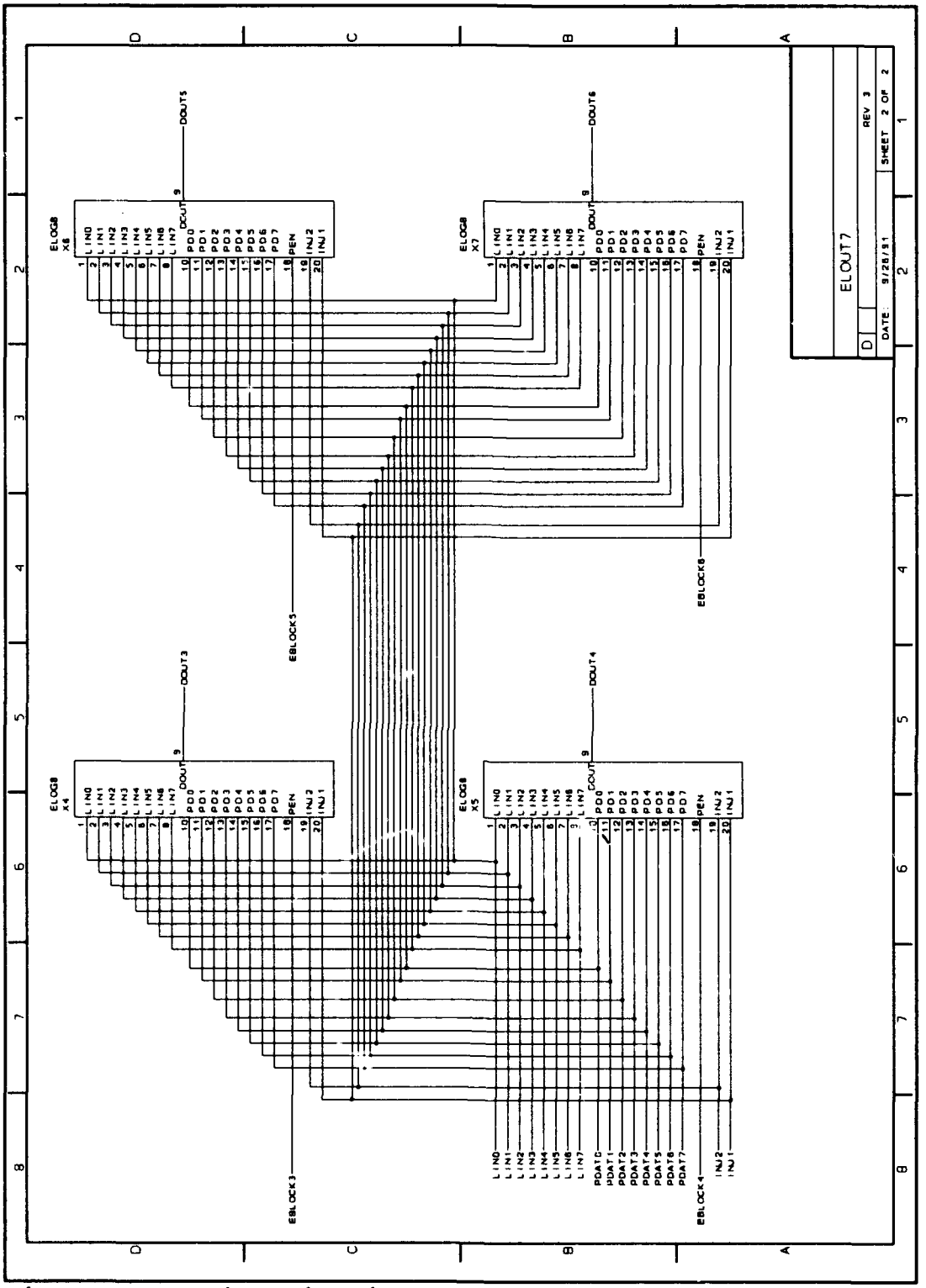


Figure 28. Microcircuit 2 Subcell ELOUT7 Schematic Diagram. Sheet 2 of 2.

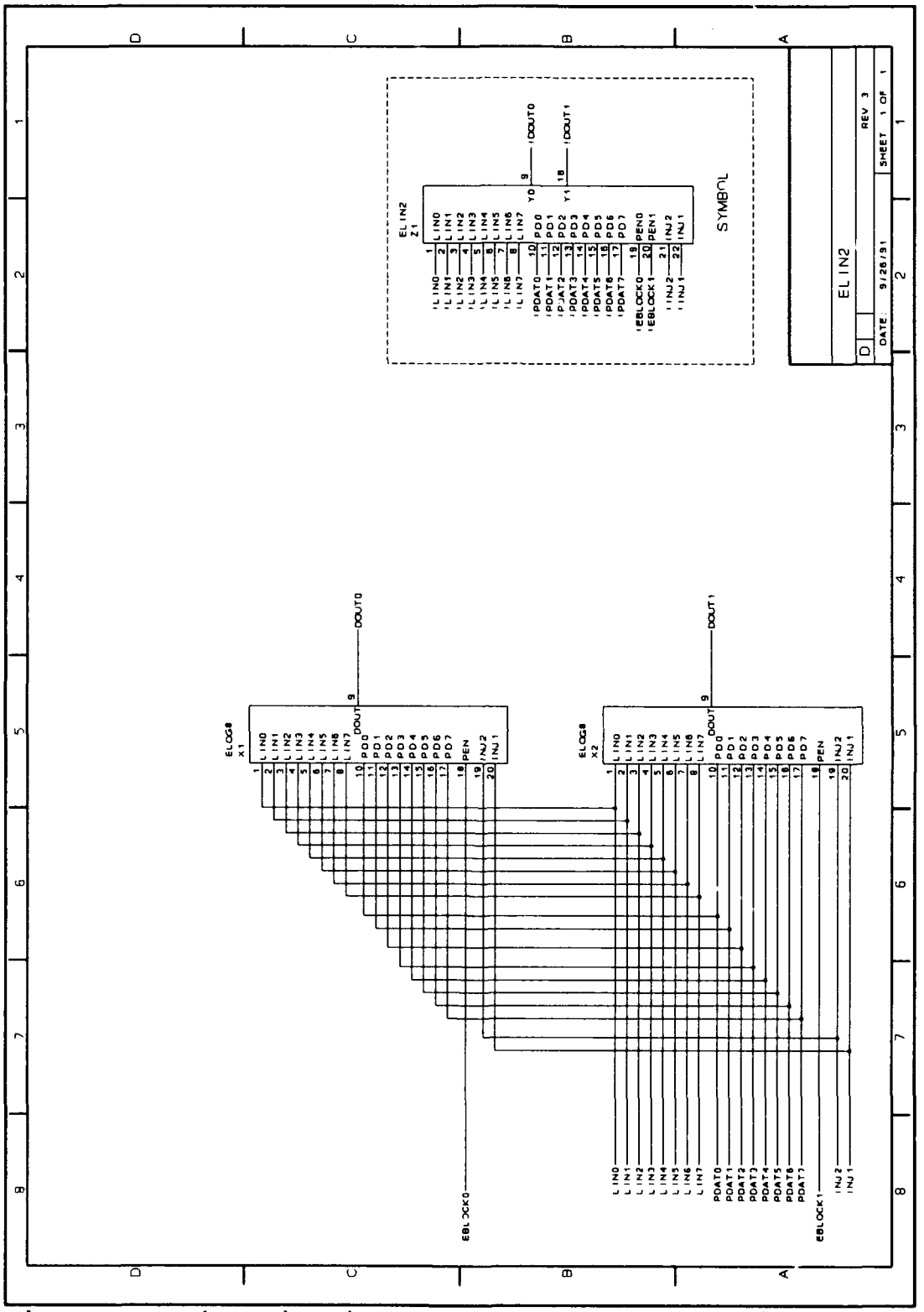


Figure 29. Microcircuit 2 Subcell ELIN2 Schematic Diagram

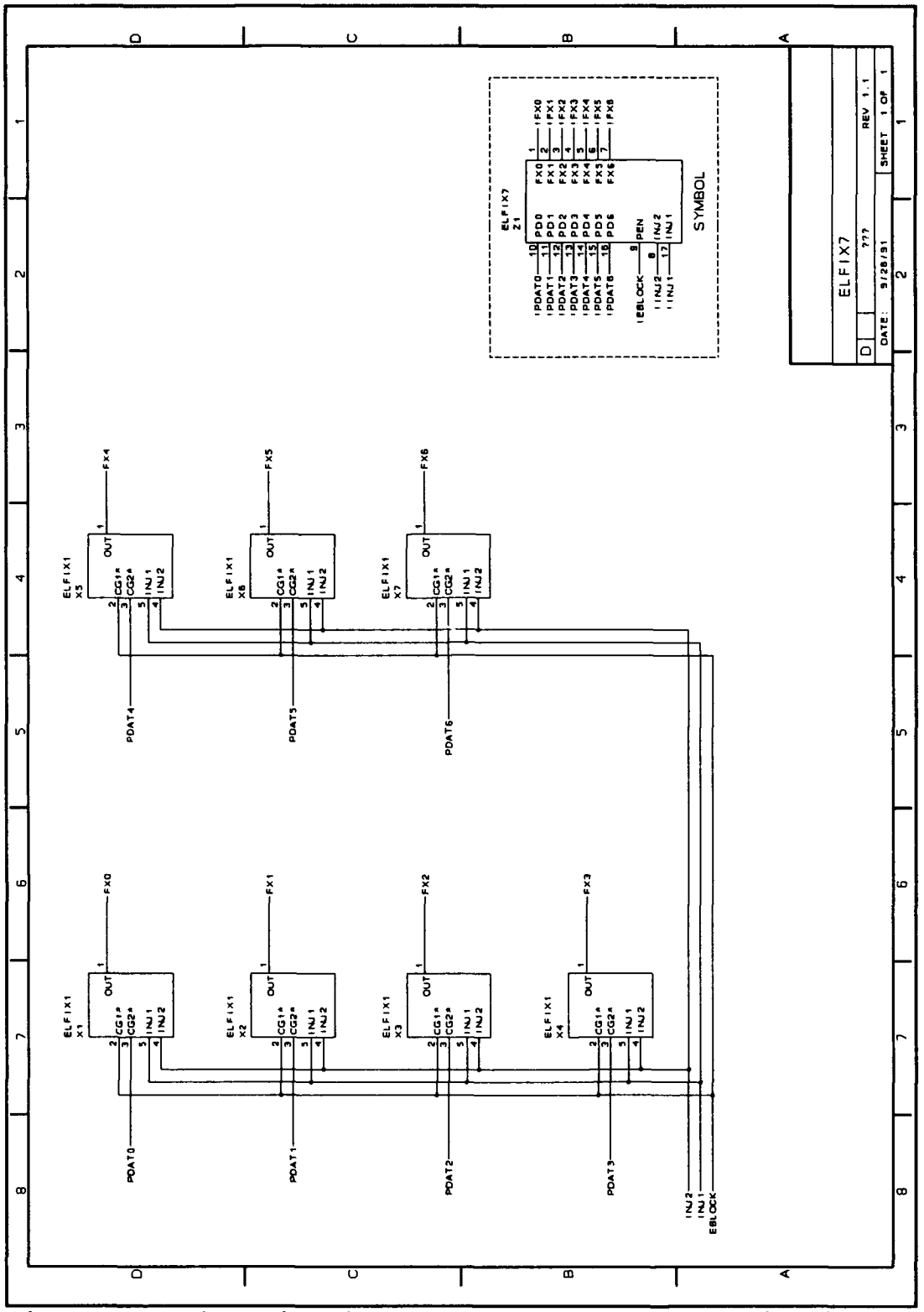


Figure 30. Microcircuit 2 Subcell ELFIX7 Schematic Diagram

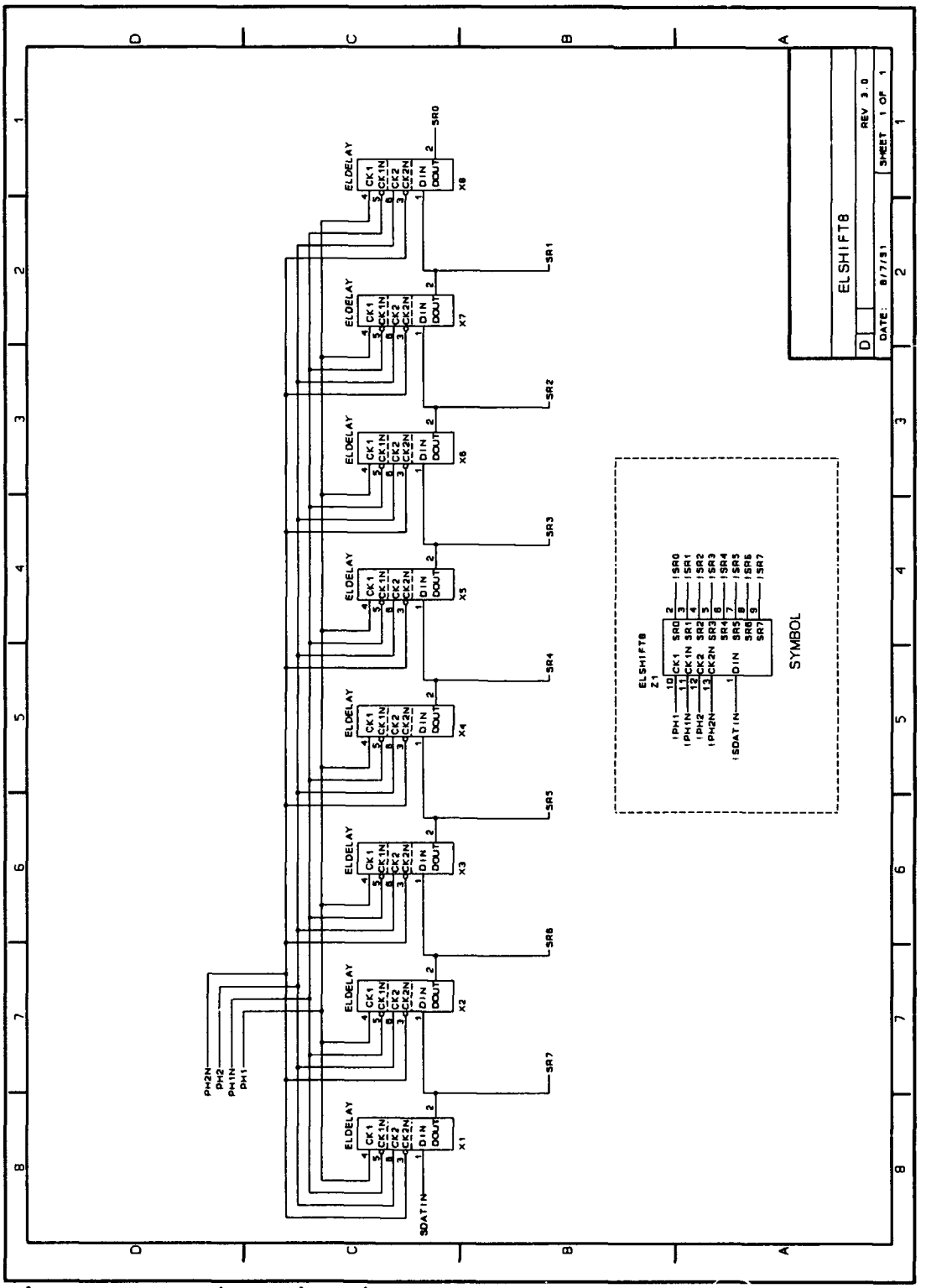


Figure 31. Microcircuit 2 Subcell ELSHIFT8 Schematic Diagram

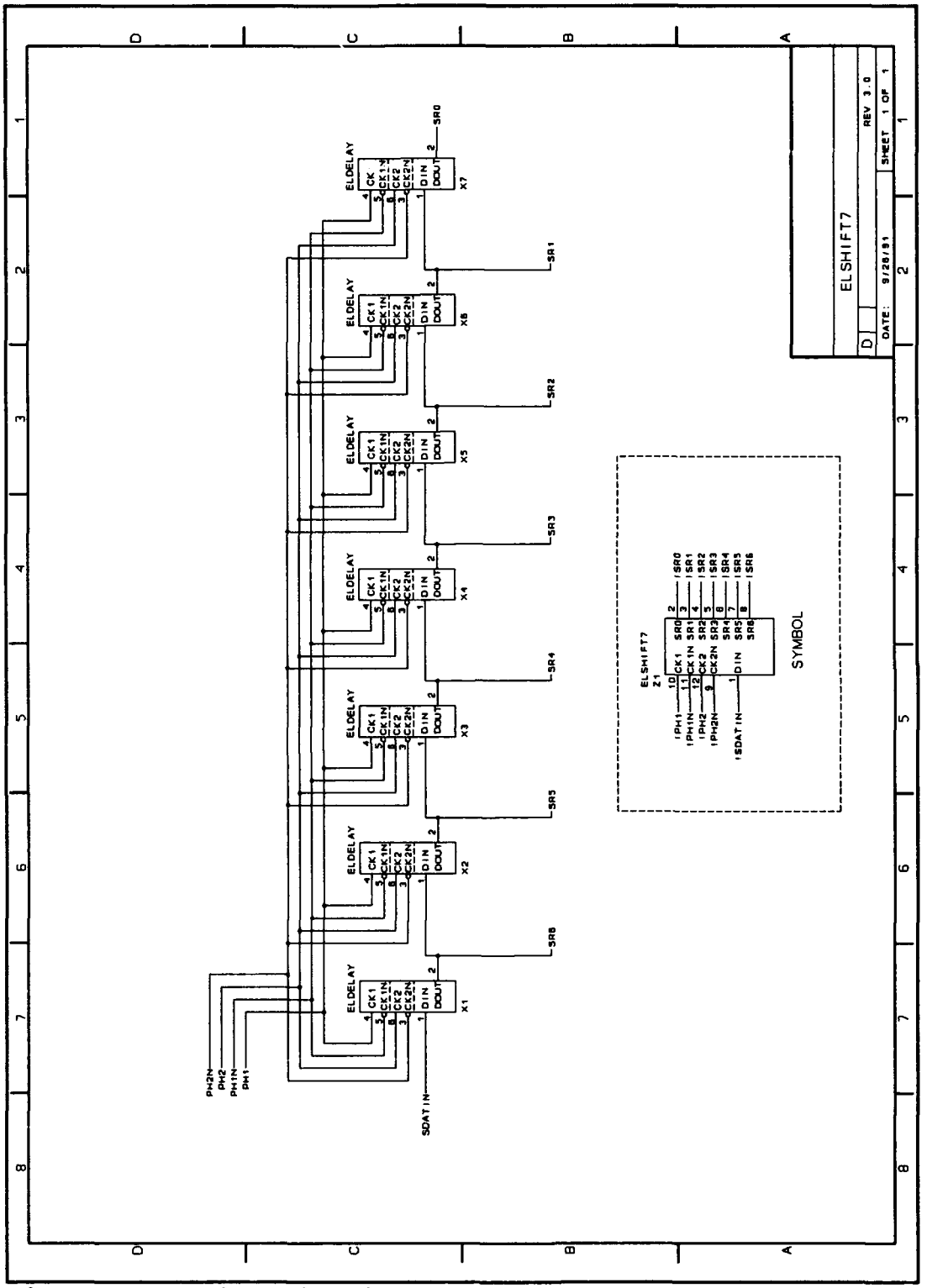


Figure 32. Microcircuit 2 Subcell ELSHIFT7 Schematic Diagram

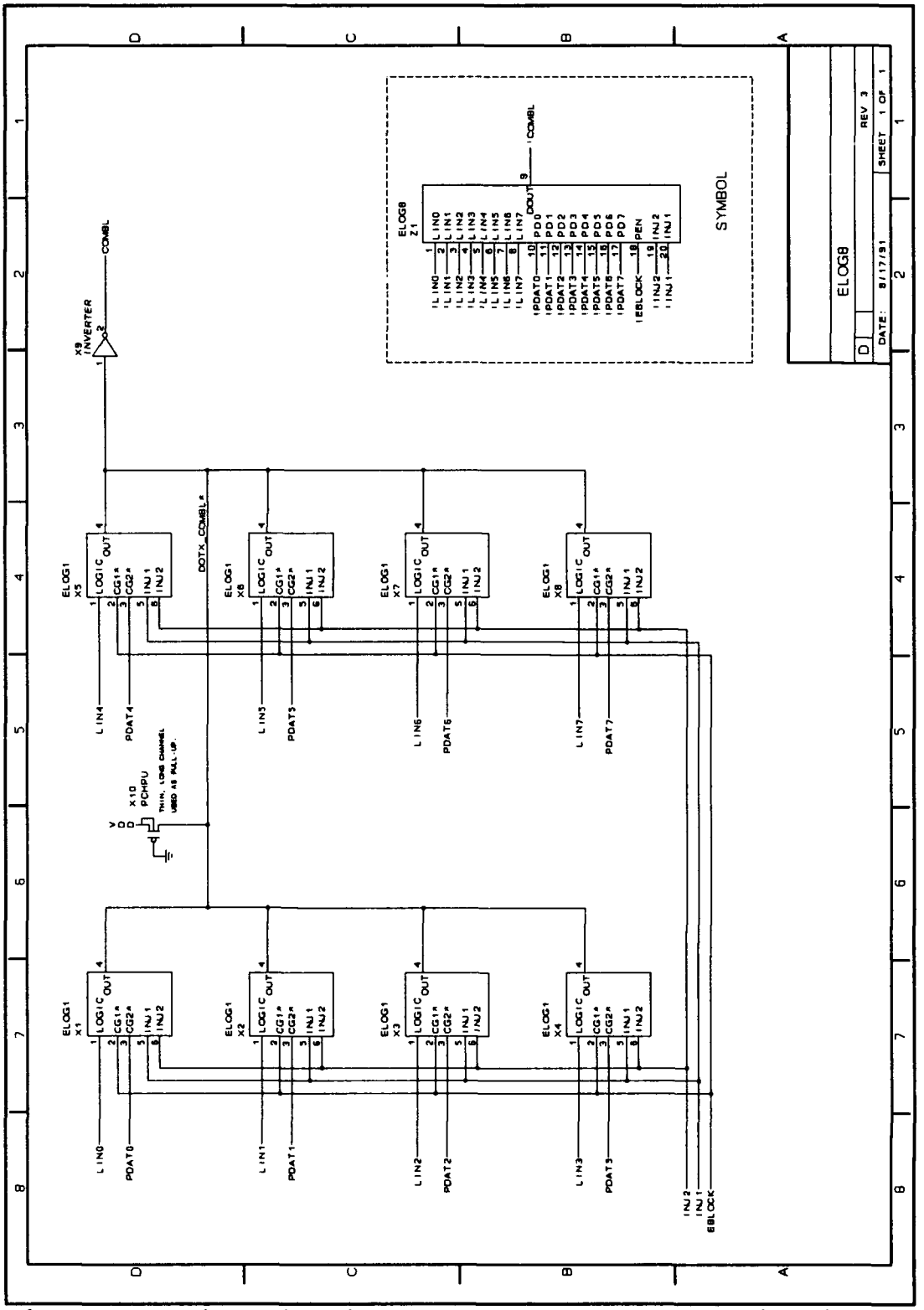


Figure 33. Microcircuit 2 Subcell ELOG8 Schematic Diagram

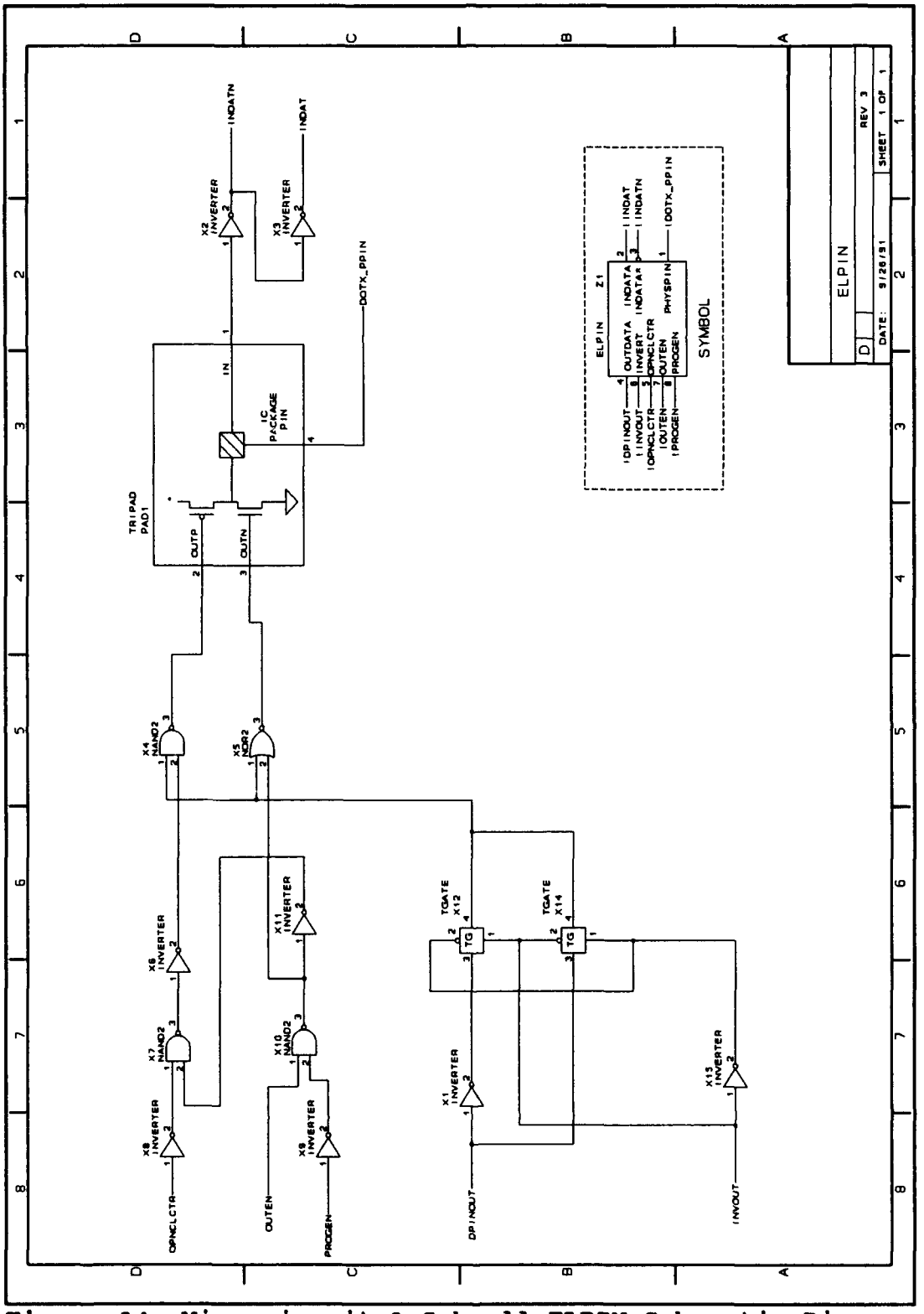


Figure 34. Microcircuit 2 Subcell ELPIN Schematic Diagram

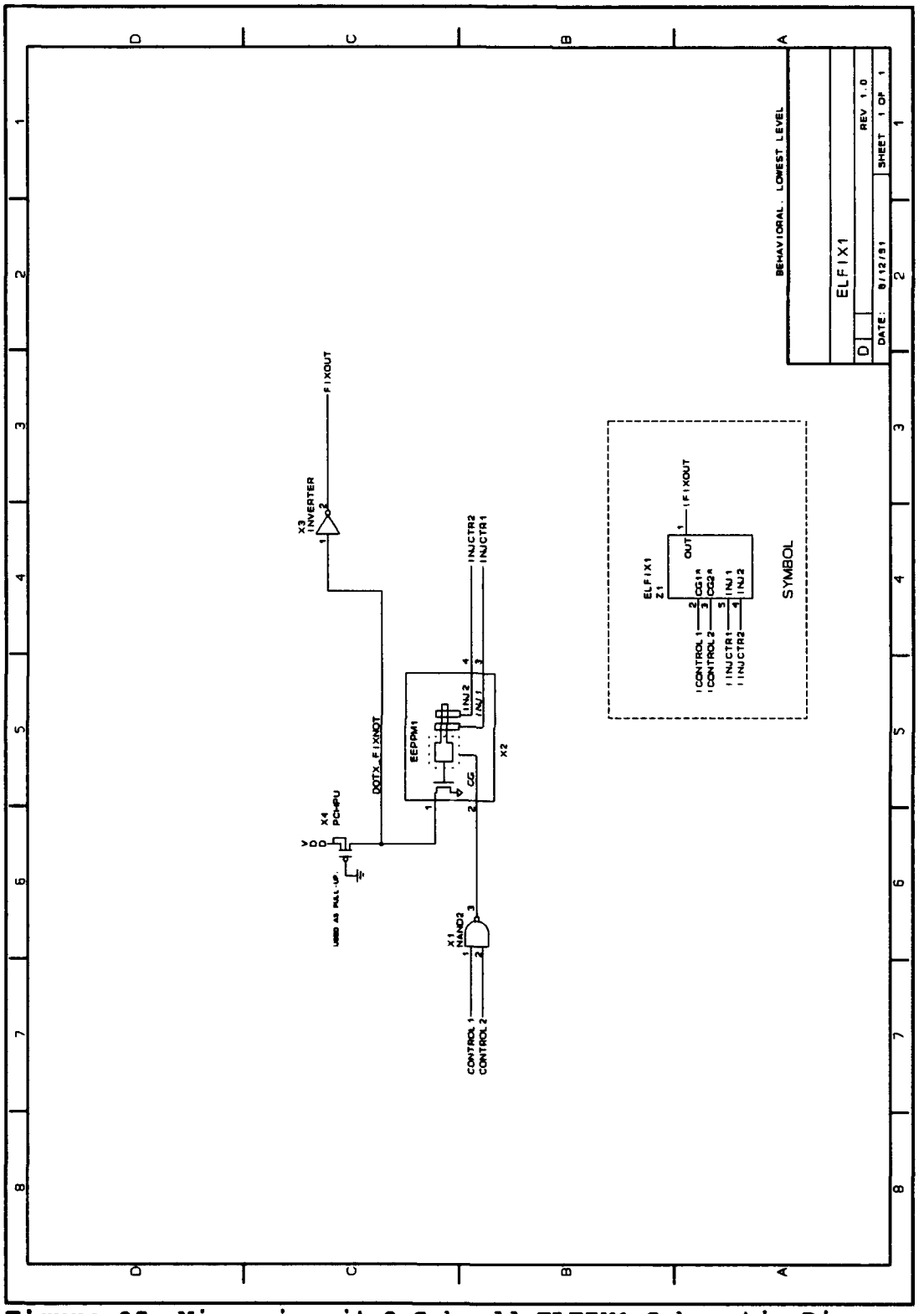


Figure 35. Microcircuit 2 Subcell ELFIX1 Schematic Diagram

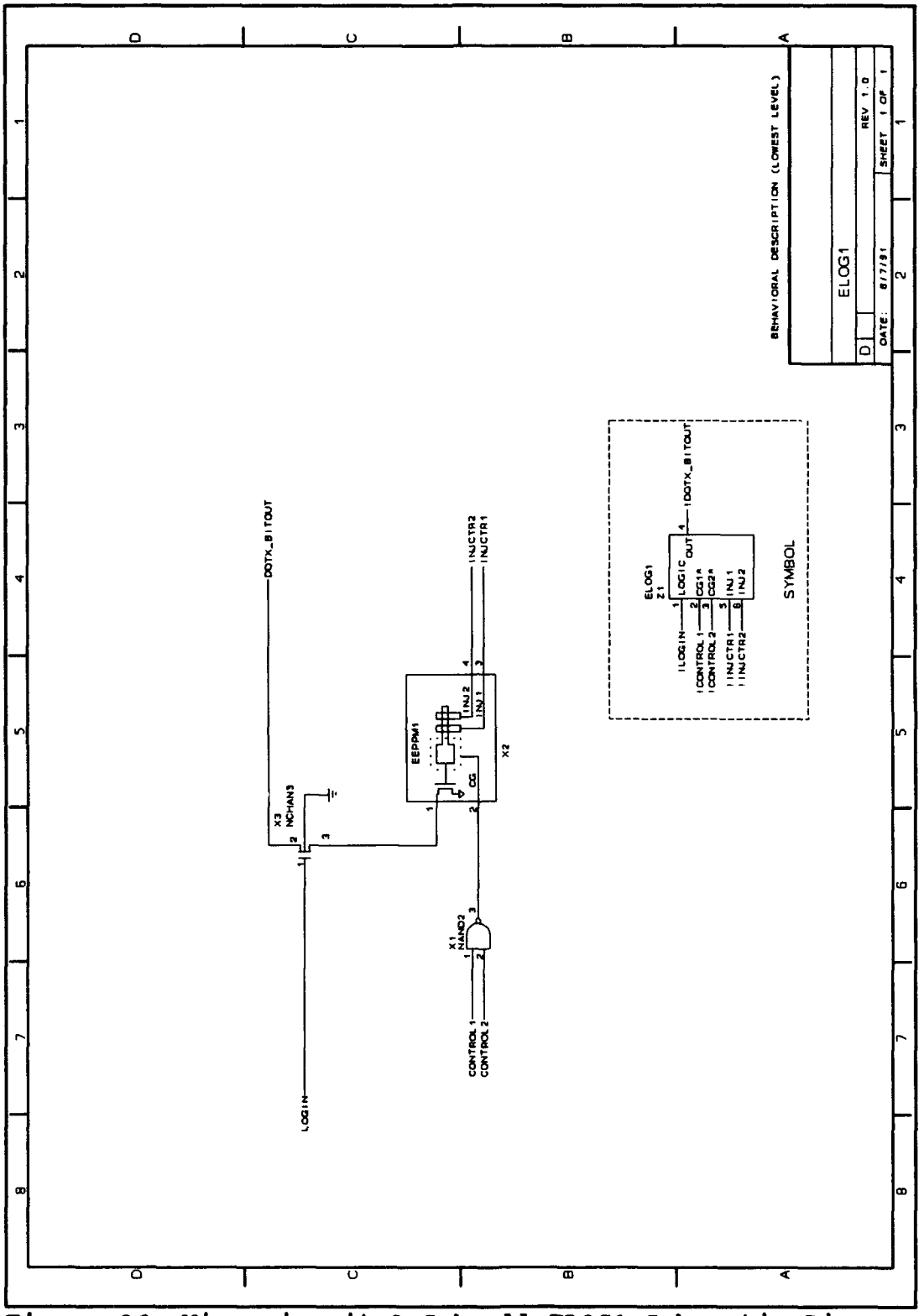


Figure 36. Microcircuit 2 Subcell ELOG1 Schematic Diagram

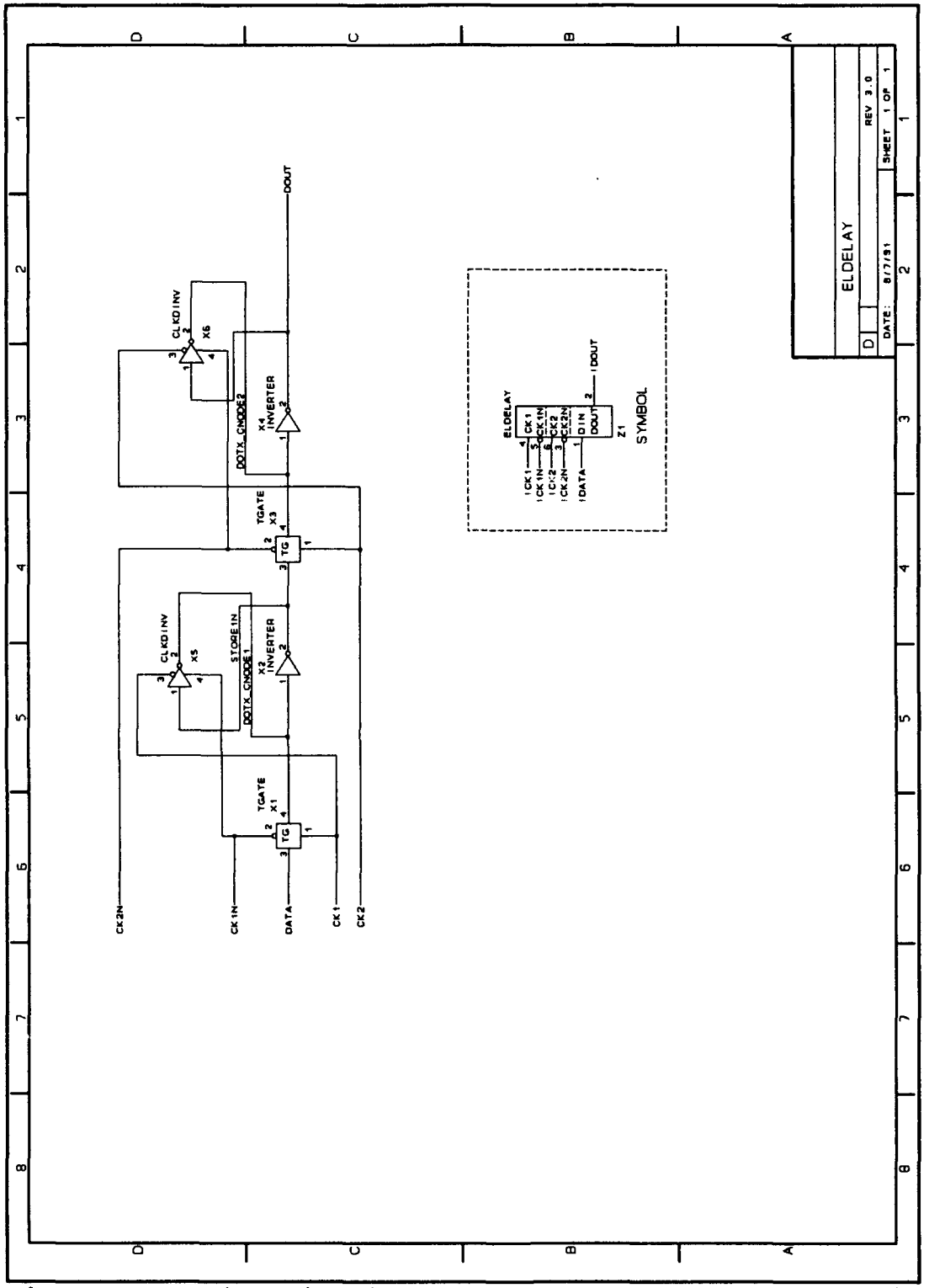


Figure 37. Microcircuit 2 Subcell ELDELAY Schematic Diagram

Appendix G: VHDL Models for Microcircuit 2

Test Bench CH2BNCH VHDL Model

```
-- Date: 10 Nov 91
-- Version:  8
--
-- Unix filename:  ch2bnch.vhd
--
-- This is a test bench for module ELPROJ
-- The framework was generated by AVG, then customized
-- by adding the process and configuration sections
-------------------------------------------------------------
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity CH2BNCH is
end CH2BNCH;

architecture A of CH2BNCH is

-- SIGNAL DECLARATIONS HERE
        signal GND : MVL7 :='0';
        signal VDD : MVL7 :='1';
    signal  LEFTA : DotX;
    signal  LEFTB : DotX;
    signal  LEFTC : DotX;
    signal  LEFTD : DotX;
    signal  LEFTE : DotX;
    signal  LEFTF : DotX;
    signal  LEFTG : DotX;
    signal  RIGHTG : DotX;
    signal  RIGHTF : DotX;
    signal  RIGHTE : DotX;
    signal  RIGHTD : DotX;
    signal  RIGHTC : DotX;
    signal  RIGHTB : DotX;
    signal  RIGHTA : DotX;
    signal  OPERATE : MVL7;
    signal  SDOUT : MVL7;
    signal  INJ1 : MVL7;
    signal  INJ2 : MVL7;
    signal TAG0 : bit;
    signal TAG1 : bit;
    signal TAG2 : bit;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELPROJ(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--           Generated on 10-07-1991 at 17:27:22
--
component ELPROJ     port(
PINAL: inout DotX;
PINBL: inout DotX;
PINCL: inout DotX;
PINDL: inout DotX;
PINEL: inout DotX;
PINFL: inout DotX;
PINGL: inout DotX;
PINGR: inout DotX;
PINFR: inout DotX;
PINER: inout DotX;
PINDR: inout DotX;
PINCR: inout DotX;
```

```
PINBR: inout DotX;
PINAR: inout DotX;
OPERATE: in MVL7;
SDOUTN: out MVL7;
INJ1: in MVL7;
INJ2: in MVL7);
end component;


begin


X1: ELPROJ PORT MAP ( LEFTA, LEFTB, LEFTC, LEFTD, LEFTE,
                      LEFTF, LEFTG, RIGHTG, RIGHTF, RIGHTE,
                      RIGHTD, RIGHTC, RIGHTB, RIGHTA, OPERATE,
                      SDOUT, INJ1, INJ2);

DRIVE: Process
procedure tick is
begin
RIGHTC <= '1';
wait for 20 ns;
RIGHTC <= '0';
RIGHTA <= '1';
wait for 20 NS;
RIGHTA <= '0';
wait for 20 NS;
end tick;

procedure ERASE is
begin
-- ERASE CYCLE
OPERATE <= '0';  -- Go to Program mode
RIGHTB <= '1';   -- Internal signals are inverted.
RIGHTC <= '1';   -- Phase 1 clock true
RIGHTA <= '1';   -- Phase 2 clock true
-- Both clocks true. All floating gate transistor control gates are high.
wait for 250 NS;
INJ1 <= '1';
wait for 50 NS;
INJ1 <= '0';
RIGHTC <= '0';
RIGHTA <= '0';
wait for 50 NS;
-- Erase Operation Complete
end ERASE;  -- Left in Program mode (Operate is false).

procedure PROGRAM is
begin
WAIT FOR 100 ns;
INJ2 <= '1';
WAIT FOR 100 ns;
INJ2 <= '0';
WAIT FOR 100 ns;
END program;

procedure SWRITE ( slct : in integer; gdat : in integer) is
variable temvar, temdat, indx : integer;
variable temvas: MVL7;
-- Will load 36 bit shift register with a 1 in the bit position of slct,
--  and the last 8 bits will be the GDAT data pattern.
begin
  temdat := gdat;
  indx := 0;
Loop1:
-- This loop takes the input data and shifts 1's into the S-R except
-- for the bit corresponding to bit position SLCT which will be zero.
```

```
  loop
    if slct = indx then temvas := '0';
      else temvas := '1';
    end if;
    RIGHTB <= temvas;
    tick;
    indx := indx + 1;
    exit loop1 when indx > 27;
  end loop loop1;
Loop2:
-- This loop takes the GDAT input data and shifts the
-- complement into the register
  for indx in 0 to 7 loop
    temvar:= temdat rem 2;
    temdat := temdat / 2;
    if temvar = 1 then temvas := '0';
      else temvas := '1';
    end if;
    RIGHTB <= temvas;
    tick;
  end loop loop2;
end SWRITE;
procedure DISCNCT is
--  Remove all logic emulation pin drive signals.
begin
  RIGHTA <= 'Z';
  RIGHTB <= 'Z';
  RIGHTC <= 'Z';
  RIGHTD <= 'Z';
  RIGHTE <= 'Z';
  RIGHTF <= 'Z';
  RIGHTG <= 'Z';
  LEFTA <= 'Z';
  LEFTB <= 'Z';
  LEFTC <= 'Z';
  LEFTD <= 'Z';
  LEFTE <= 'Z';
  LEFTF <= 'Z';
  LEFTG <= 'Z';
end DISCNCT;


-- Main Test
begin
-- Initialize
   INJ1 <= '0';
   INJ2 <= '0';
   OPERATE <= '0';
   RIGHTC <= '0';  -- Phase 1 clock false
   RIGHTA <= '0';  -- Phase 2 clock false
   RIGHTB <= '1';  -- Serial Data Input
   wait for 100 NS;
   tick;
   ERASE;
   TAG0 <= '0';
   TAG1 <= '0';
   TAG2 <= '0';
--  --  --  --  --  --  --  --  --  --  --  --  --
                      -- Program for 7400 emulation
   SWRITE(2,2);
   Program;
   SWRITE(4,16);
   Program;
   SWRITE(8,8);
   Program;
   SWRITE(13,64);
   Program;
```

```
    SWRITE(15,18);
    Program;
    SWRITE(18,80);
    Program;
    SWRITE(20,80);
    Program;
    SWRITE(22,18);
    Program;
    SWRITE(25,80);
    Program;
    SWRITE(27,80);
    Program;
    DISCNCT;
    Operate <= '1';
    -- Finished programming for 7400
    wait for 100 NS;
    TAG0 <= '1';
    LEFTA <= '0';
    LEFTB <= '0';
    LEFTD <= '0';
    LEFTE <= '0';
    RIGHTA <= '0';
    RIGHTB <= '0';
    RIGHTD <= '0';
    RIGHTE <= '0';
    wait for 100 NS;
    LEFTA <= '1';
    LEFTD <= '1';
    RIGHTA <= '1';
    RIGHTD <= '1';
    wait for 100 NS;
    LEFTA <= '0';
    LEFTD <= '0';
    RIGHTA <= '0';
    RIGHTD <= '0';
    LEFTB <= '1';
    LEFTE <= '1';
    RIGHTB <= '1';
    RIGHTE <= '1';
    wait for 100 NS;
    LEFTA <= '1';
    LEFTD <= '1';
    RIGHTA <= '1';
    RIGHTD <= '1';
    wait for 100 NS;
    DISCNCT;
--  --  --  --  --  --  --  --  --  --  --  --  --
  erase;
--  --  --  --  --  --  --  --  --  --  --  --  --
                      -- Program for 7403 emulation
    SWRITE(2,2);
    Program;
    SWRITE(4,16);
    Program;
    SWRITE(8,8);
    Program;
    SWRITE(13,64);
    Program;
    SWRITE(14,18);
    Program;
    SWRITE(15,18);
    Program;
    SWRITE(18,80);
    Program;
    SWRITE(20,80);
    Program;
    SWRITE(21,18);
```

```
    Program;
    SWRITE(22,18);
    Program;
    SWRITE(25,80);
    Program;
    SWRITE(27,80);
    Program;
    DISCNCT;
    Operate <= '1';
    -- Finished programming
    wait for 100 NS;
    TAG0 <= '0';
    TAG1 <= '1';
    LEFTA <= '0';
    LEFTB <= '0';
    LEFTD <= '0';
    LEFTE <= '0';
    RIGHTA <= '0';
    RIGHTB <= '0';
    RIGHTD <= '0';
    RIGHTE <= '0';
    wait for 100 NS;
    LEFTA <= '1';
    LEFTD <= '1';
    RIGHTA <= '1';
    RIGHTD <= '1';
    wait for 100 NS;
    LEFTA <= '0';
    LEFTD <= '0';
    RIGHTA <= '0';
    RIGHTD <= '0';
    LEFTB <= '1';
    LEFTE <= '1';
    RIGHTB <= '1';
    RIGHTE <= '1';
    wait for 100 NS;
    LEFTA <= '1';
    LEFTD <= '1';
    RIGHTA <= '1';
    RIGHTD <= '1';
    wait for 100 NS;
    DISCNCT;
--  --  --  --  --  --  --  --  --  --  --  --  --  --
  erase;
--  --  --  --  --  --  --  --  --  --  --  --  --  --
                      -- Program for 7404 emulation
    SWRITE(0,8);
    Program;
    SWRITE(1,4);
    Program;
    SWRITE(2,2);
    Program;
    SWRITE(13,64);
    Program;
    SWRITE(12,32);
    Program;
    SWRITE(11,16);
    Program;
    SWRITE(15,42);
    Program;
    SWRITE(16,42);
    Program;
    SWRITE(18,32);
    Program;
    SWRITE(19,8);
    Program;
    SWRITE(20,128);
```

```
    Program;
    SWRITE(22,42);
    Program;
    SWRITE(23,42);
    Program;
    SWRITE(25,32);
    Program;
    SWRITE(26,8);
    Program;
    SWRITE(27,128);
    Program;
    DISCNCT;
    Operate <= '1';
    -- Finished programming
    wait for 100 NS;
    TAG0 <= '1';
    TAG1 <= '1';
    TAG2 <= '0';
    LEFTA <= '0';
    LEFTC <= '0';
    LEFTE <= '0';
    RIGHTA <= '0';
    RIGHTC <= '0';
    RIGHTE <= '0';
    wait for 100 NS;
    LEFTA <= '1';
    wait for 100 NS;
    LEFTC <= '1';
    wait for 100 NS;
    LEFTE <= '1';
    wait for 100 NS;
    RIGHTA <= '1';
    wait for 100 NS;
    RIGHTC <= '1';
    wait for 100 NS;
    RIGHTE <= '1';
    wait for 100 NS;
    DISCNCT;
--  --  --  --  --  --  --  --  --  --  --  --  --

  erase;
--  --  --  --  --  --  --  --  --  --  --  --  --
                      -- Program for 7408 emulation
    SWRITE(2,2);
    Program;
    SWRITE(4,16);
    Program;
    SWRITE(8,8);
    Program;
    SWRITE(13,64);
    Program;
    SWRITE(15,18);
    Program;
    SWRITE(16,18);
    Program;
    SWRITE(18,80);
    Program;
    SWRITE(20,80);
    Program;
    SWRITE(22,18);
    Program;
    SWRITE(23,18);
    Program;
    SWRITE(25,80);
    Program;
    SWRITE(27,80);
    Program;
```

```
    DISCNCT;
    Operate <= '1';
    -- Finished programming
    wait for 100 NS;
    TAG0 <= '0';
    TAG1 <= '0';
    TAG2 <= '1';
    LEFTA <= '0';
    LEFTB <= '0';
    LEFTD <= '0';
    LEFTE <= '0';
    RIGHTA <= '0';
    RIGHTB <= '0';
    RIGHTD <= '0';
    RIGHTE <= '0';
    wait for 100 NS;
    LEFTA <= '1';
    LEFTD <= '1';
    RIGHTA <= '1';
    RIGHTD <= '1';
    wait for 100 NS;
    LEFTA <= '0';
    LEFTD <= '0';
    RIGHTA <= '0';
    RIGHTD <= '0';
    LEFTB <= '1';
    LEFTE <= '1';
    RIGHTB <= '1';
    RIGHTE <= '1';
    wait for 100 NS;
    LEFTA <= '1';
    LEFTD <= '1';
    RIGHTA <= '1';
    RIGHTD <= '1';
    wait for 100 NS;
    DISCNCT;
--  --  --  --  --  --  --  --  --  --  --  --  --  --
  erase;
--  --  --  --  --  --  --  --  --  --  --  --  --  --
                        -- Program for 7420 emulation
    SWRITE(2,10);
    Program;
    SWRITE(13,80);
    Program;
    SWRITE(15,2);
    Program;
    SWRITE(18,80);
    Program;
    SWRITE(20,80);
    Program;
    SWRITE(22,2);
    Program;
    SWRITE(25,80);
    Program;
    SWRITE(27,80);
    Program;
    DISCNCT;
    Operate <= '1';
    -- Finished programming
    wait for 100 NS;
    TAG0 <= '1';
    TAG1 <= '0';
    TAG2 <= '1';
    LEFTA <= '0';
    LEFTB <= '0';
    LEFTD <= '0';
    LEFTE <= '0';
```

```
RIGHTA <= '0';
RIGHTB <= '0';
RIGHTD <= '0';
RIGHTE <= '0';
wait for 100 NS;  --  E D B A = 0000
LEFTA <= '1';
RIGHTA <= '1';
wait for 100 NS;  -- 0001
LEFTA <= '0';
RIGHTA <= '0';
LEFTB <= '1';
RIGHTB <= '1';
wait for 100 NS;   -- 0010
LEFTA <= '1';
RIGHTA <= '1';
wait for 100 NS;   -- 0011
LEFTD <= '1';
RIGHTD <= '1';
LEFTA <= '0';
LEFTB <= '0';
RIGHTA <= '0';
RIGHTB <= '0';
wait for 100 NS;   -- 0100
LEFTA <= '1';
RIGHTA <= '1';
wait for 100 NS;  -- 0101
LEFTA <= '0';
RIGHTA <= '0';
LEFTB <= '1';
RIGHTB <= '1';
wait for 100 NS;  -- 0110
LEFTA <= '1';
RIGHTA <= '1';
wait for 100 NS; -- 0111
LEFTD <= '0';
RIGHTD <= '0';
LEFTA <= '0';
LEFTB <= '0';
RIGHTA <= '0';
RIGHTB <= '0';
LEFTE <= '1';
RIGHTE <= '1';
wait for 100 NS;  -- 1000
LEFTA <= '1';
RIGHTA <= '1';
wait for 100 NS;  -- 1001
LEFTA <= '0';
RIGHTA <= '0';
LEFTB <= '1';
RIGHTB <= '1';
wait for 100 NS;  --1010
LEFTA <= '1';
RIGHTA <= '1';
wait for 100 NS;  -- 1011
LEFTD <= '1';
RIGHTD <= '1';
LEFTA <= '0';
LEFTB <= '0';
RIGHTA <= '0';
RIGHTB <= '0';
wait for 100 NS;  -- 1100
LEFTA <= '1';
RIGHTA <= '1';
wait for 100 NS;  -- 1101
LEFTA <= '0';
RIGHTA <= '0';
LEFTB <= '1';
```

```
   RIGHTB <= '1';
   wait for 100 NS;  -- 1110
   LEFTA <= '1';
   RIGHTA <= '1';
   wait for 100 NS;  -- 1111
   DISCNCT;
--  --  --  --  --  --  --  --  --  --  --  --  --

  erase;
--  --  --  --  --  --  --  --  --  --  --  --  --
                     -- Program for 7427 emulation
   SWRITE(2,6);
   Program;
   SWRITE(13,96);
   Program;
   SWRITE(11,24);
   Program;
   SWRITE(15,2);
   Program;
   SWRITE(16,2);
   Program;
   SWRITE(18,32);
   Program;
   SWRITE(19,10);
   Program;
   SWRITE(20,160);
   Program;
   SWRITE(22,34);
   Program;
   SWRITE(23,34);
   Program;
   SWRITE(25,32);
   Program;
   SWRITE(26,10);
   Program;
   SWRITE(27,128);
   Program;
   DISCNCT;
   Operate <= '1';
   -- Finished programming
   wait for 100 NS;
   TAG0 <= '0';
   TAG1 <= '1';
   TAG2 <= '1';

   LEFTA <= '0';
   LEFTC <= '0';
   RIGHTC <= '0';
--
   LEFTB <= '0';
   LEFTD <= '0';
   RIGHTD <= '0';
--
   RIGHTA <= '0';
   LEFTE <= '0';
   RIGHTE <= '0';
   wait for 100 NS;
   LEFTA <= '1';
   LEFTC <= '1';
   RIGHTC <= '1';
   wait for 100 NS;
   LEFTA <= '0';
   LEFTC <= '0';
   RIGHTC <= '0';
--
   LEFTB <= '1';
   LEFTD <= '1';
```

```
   RIGHTD <= '1';
   wait for 100 NS;
   LEFTA <= '1';
   LEFTC <= '1';
   RIGHTC <= '1';
   wait for 100 NS;
   LEFTA <= '0';
   LEFTC <= '0';
   RIGHTC <= '0';
--
   LEFTB <= '0';
   LEFTD <= '0';
   RIGHTD <= '0';
--
   RIGHTA <= '1';
   LEFTE <= '1';
   RIGHTE <= '1';
   wait for 100 NS;
   LEFTA <= '1';
   LEFTC <= '1';
   RIGHTC <= '1';
   wait for 100 NS;
   LEFTA <= '0';
   LEFTC <= '0';
   RIGHTC <= '0';
--
   LEFTB <= '1';
   LEFTD <= '1';
   RIGHTD <= '1';
   wait for 100 NS;
   LEFTA <= '1';
   LEFTC <= '1';
   RIGHTC <= '1';
   wait for 100 NS;
   DISCNCT;
--  --  --  --  --  --  --  --  --  --  --  --  --  --
   erase;
--  --  --  --  --  --  --  --  --  --  --  --  --  --
                        -- Program for 74S133 emulation
   SWRITE(6,90);
   Program;
   SWRITE(18,85);
   Program;
   SWRITE(20,85);
   Program;
   SWRITE(22,1);
   Program;
   SWRITE(25,84);
   Program;
   SWRITE(27,85);
   Program;
   DISCNCT;
   Operate <= '1';
   -- Finished programming
   wait for 100 NS;
   TAG0 <= '1';
   TAG1 <= '1';
   TAG2 <= '1';
   LEFTA <= '0';
   LEFTB <= '0';
   LEFTC <= '0';
   LEFTD <= '0';
   LEFTE <= '0';
   LEFTF <= '0';
   LEFTG <= '0';
   RIGHTA <= '0';
   RIGHTB <= '0';
```

```
   RIGHTC <= '0';
   RIGHTD <= '0';
   RIGHTE <= '0';
   RIGHTF <= '0';
   wait for 100 NS;
   LEFTA <= '1';
   wait for 100 NS;
   LEFTB <= '1';
   wait for 100 NS;
   LEFTC <= '1';
   wait for 100 NS;
   LEFTD <= '1';
   wait for 100 NS;
   LEFTE <= '1';
   wait for 100 NS;
   LEFTF <= '1';
   wait for 100 NS;
   LEFTG <= '1';
   wait for 100 NS;
   RIGHTA <= '1';
   wait for 100 NS;
   RIGHTB <= '1';
   wait for 100 NS;
   RIGHTC <= '1';
   wait for 100 NS;
   RIGHTD <= '1';
   wait for 100 NS;
   RIGHTE <= '1';
   wait for 100 NS;
   RIGHTF <= '1';
   wait for 100 NS;
   DISCNCT;
-- -- -- -- -- -- -- -- -- -- -- -- --

   wait;
end process;

end A;

configuration CTEST of CH2BNCH is
   for A
          for all : ELPROJ
                        use entity WORK.ELPROJ(structure);
          end for;
      end for;
end CTEST;
```

Top Level ELPROJ VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELPROJ(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:37:14
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELPROJ is
     port(
PINAL: inout DotX;
PINBL: inout DotX;
PINCL: inout DotX;
PINDL: inout DotX;
PINEL: inout DotX;
PINFL: inout DotX;
PINGL: inout DotX;
PINGR: inout DotX;
PINFR: inout DotX;
PINER: inout DotX;
PINDR: inout DotX;
PINCR: inout DotX;
PINBR: inout DotX;
PINAR: inout DotX;
OPERATE: in MVL7;
SDOUTN: out MVL7;
INJ1: in MVL7;
INJ2: in MVL7 );
end ELPROJ;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELPROJ is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  S02028035 : MVL7;
    signal  INLB : MVL7;
    signal  S01018035 : MVL7;
    signal  PROGEN : MVL7;
    signal  S01017018 : MVL7;
    signal  PH1 : MVL7;
    signal  PH1N : MVL7;
    signal  S01018023 : MVL7;
    signal  PH2 : MVL7;
    signal  PH2N : MVL7;
    signal  S01027034 : MVL7;
    signal  SDIN : MVL7;
    signal  INLA : MVL7;
    signal  LTERM0 : MVL7;
    signal  LTERM1 : MVL7;
    signal  LTERM2 : MVL7;
    signal  LTERM3 : MVL7;
    signal  S02052024 : MVL7;
    signal  S02020061 : MVL7;
    signal  GPW7 : MVL7;
    signal  GPW6 : MVL7;
    signal  GPW5 : MVL7;
    signal  GPW4 : MVL7;
    signal  GPW3 : MVL7;
    signal  GPW2 : MVL7;
    signal  GPW1 : MVL7;
```

```
    signal  GPWO : MVL7;
    signal  LGOUPAL : MVL7;
    signal  LGOUPBL : MVL7;
    signal  LGOUPCL : MVL7;
    signal  LGOUPDL : MVL7;
    signal  LGOUPEL : MVL7;
    signal  LGOUPFL : MVL7;
    signal  LGOUPGL : MVL7;
    signal  INLC : MVL7;
    signal  RTERM0 : MVL7;
    signal  RTERM1 : MVL7;
    signal  RTERM2 : MVL7;
    signal  RTERM3 : MVL7;
    signal  LGOUPAR : MVL7;
    signal  LGOUPBR : MVL7;
    signal  LGOUPCR : MVL7;
    signal  LGOUPDR : MVL7;
    signal  LGOUPER : MVL7;
    signal  LGOUPFR : MVL7;
    signal  LGOUPGR : MVL7;
    signal  S02028036 : MVL7;
    signal  S02028037 : MVL7;
    signal  S02028038 : MVL7;
    signal  S02028039 : MVL7;
    signal  S02028040 : MVL7;
    signal  S02028041 : MVL7;
    signal  S02028042 : MVL7;
    signal  S02028043 : MVL7;
    signal  S02028044 : MVL7;
    signal  S02028045 : MVL7;
    signal  S02028046 : MVL7;
    signal  S02028047 : MVL7;
    signal  S02028048 : MVL7;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

component inverter
  port(input: in MVL7; output: out MVL7);
end component;
component nand2
  port(ND2IN1: in MVL7; ND2IN2 : in MVL7; ND2OUT: out MVL7);
end component;
-- COMPONENT GENERATED BY AVG FROM FILE ELPRO7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:37:04
--
component ELPRO7     port(
PPINA: inout DotX;
PPINB: inout DotX;
PPINC: inout DotX;
PPIND: inout DotX;
PPINE: inout DotX;
PPINF: inout DotX;
PPING: inout DotX;
LOG0: out MVL7;
LOG1: out MVL7;
LOG2: out MVL7;
LOG3: out MVL7;
SDIN: in MVL7;
CSEL0: inout MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
PH2N: in MVL7;
PROGEN: in MVL7;
```

```
INJ1: in MVL7;
INJ2: in MVL7;
GDAT0: in MVL7;
GDAT1: in MVL7;
GDAT2: in MVL7;
GDAT3: in MVL7;
GDAT4: in MVL7;
GDAT5: in MVL7;
GDAT6: in MVL7;
GDAT7: in MVL7;
OUTPA: in MVL7;
OUTPB: in MVL7;
OUTPC: in MVL7;
OUTPD: in MVL7;
OUTPE: in MVL7;
OUTPF: in MVL7;
OUTPG: in MVL7;
INPINA: inout MVL7;
INPINB: inout MVL7;
INPINC: inout MVL7;
INPIND: inout MVL7;
INPINE: inout MVL7;
INPINF: inout MVL7;
INPING: inout MVL7);
end component;

-- COMPONENT GENERATED BY AVG FROM FILE ELOUT14(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:37:01
--
component ELOUT14     port(
LIN0: in MVL7;
LIN1: in MVL7;
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
DOUT0: out MVL7;
DOUT1: out MVL7;
DOUT2: out MVL7;
DOUT3: out MVL7;
DOUT4: out MVL7;
DOUT5: out MVL7;
DOUT6: out MVL7;
DOUT7: out MVL7;
DOUT8: out MVL7;
DOUT9: out MVL7;
DOUT10: out MVL7;
DOUT11: out MVL7;
DOUT12: out MVL7;
DOUT13: out MVL7;
EBLOCK0: in MVL7;
EBLOCK1: in MVL7;
EBLOCK2: in MVL7;
EBLOCK3: in MVL7;
EBLOCK4: in MVL7;
EBLOCK5: in MVL7;
```

```
EBLOCK6: in MVL7;
EBLOCK7: in MVL7;
EBLOCK8: in MVL7;
EBLOCK9: in MVL7;
EBLOCK10: in MVL7;
EBLOCK11: in MVL7;
EBLOCK12: in MVL7;
EBLOCK13: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7);
end component;

-- COMPONENT GENERATED BY AVG FROM FILE ELSHIFT8(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:36:52
--
component ELSHIFT8     port(
SDATIN: in MVL7;
SR0: inout MVL7;
SR1: inout MVL7;
SR2: inout MVL7;
SR3: inout MVL7;
SR4: inout MVL7;
SR5: inout MVL7;
SR6: inout MVL7;
SR7: inout MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
PH2N: in MVL7);
end component;

-- COMPONENT GENERATED BY AVG FROM FILE ELSHFT14(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:37:00
--
component ELSHFT14     port(
SDATIN: in MVL7;
SR0: inout MVL7;
SR1: inout MVL7;
SR2: inout MVL7;
SR3: inout MVL7;
SR4: inout MVL7;
SR5: inout MVL7;
SR6: inout MVL7;
SR7: inout MVL7;
SR8: inout MVL7;
SR9: inout MVL7;
SR10: inout MVL7;
SR11: inout MVL7;
SR12: inout MVL7;
SR13: inout MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
PH2N: in MVL7);
end component;


begin


P1INVXX: INVERTER PORT MAP ( S02028035, SDOUTN);

X1: INVERTER PORT MAP ( INL8, S01018035);

X10: NAND2 PORT MAP ( PROGEN, S01017018, PH1);
```

```
X11: INVERTER PORT MAP ( PH1, PH1N);

X12: NAND2 PORT MAP ( S01018023, PROGEN, PH2);

X14: INVERTER PORT MAP ( PH2, PH2N);

X15: NAND2 PORT MAP ( PROGEN, S01018035, S01027034);

X16: INVERTER PORT MAP ( S01027034, SDIN);

X2: INVERTER PORT MAP ( INLA, S01018023);

X3: ELPRO7 PORT MAP ( PINAL, PINBL, PINCL, PINDL, PINEL,
                      PINFL, PINGL, LTERM0, LTERM1, LTERM2,
                      LTERM3, S02052024, S02020061, PH1, PH1N,
                      PH2, PH2N, PROGEN, INJ1, INJ2,
                      GPW7, GPW6, GPW5, GPW4, GPW3,
                      GPW2, GPW1, GPW0, LGOUPAL, LGOUPBL,
                      LGOUPCL, LGOUPDL, LGOUPEL, LGOUPFL, LGOUPGL,
                      OPEN, OPEN, OPEN, OPEN, OPEN,
                      OPEN, OPEN);

X4: INVERTER PORT MAP ( INLC, S01017018);

X5: ELPRO7 PORT MAP ( PINAR, PINBR, PINCR, PINDR, PINER,
                      PINFR, PINGR, RTERM0, RTERM1, RTERM2,
                      RTERM3, GPW0, S02052024, PH1, PH1N,
                      PH2, PH2N, PROGEN, INJ1, INJ2,
                      GPW7, GPW6, GPW5, GPW4, GPW3,
                      GPW2, GPW1, GPW0, LGOUPAR, LGOUPBR,
                      LGOUPCR, LGOUPDR, LGOUPER, LGOUPFR, LGOUPGR,
                      INLA, INLB, INLC, OPEN, OPEN,
                      OPEN, OPEN);

X6: INVERTER PORT MAP ( OPERATE, PROGEN);

X7: ELOUT14 PORT MAP ( RTERM3, RTERM2, RTERM1, RTERM0, LTERM0,
                       LTERM1, LTERM2, LTERM3, GPW7, GPW6,
                       GPW5, GPW4, GPW3, GPW2, GPW1,
                       GPW0, LGOUPBL, LGOUPDL, LGOUPFL, LGOUPAR,
                       LGOUPCR, LGOUPER, LGOUPGR, LGOUPAL, LGOUPCL,
                       LGOUPEL, LGOUPGL, LGOUPBR, LGOUPDR, LGOUPFR,
                       S02028035, S02028036, S02028037, S02028038, S02028039,
                       S02028040, S02028041, S02028042, S02028043, S02028044,
                       S02028045, S02028046, S02028047, S02028048, INJ2,
                       INJ1);

X8: ELSHIFT8 PORT MAP ( SDIN, GPW0, GPW1, GPW2, GPW3,
                        GPW4, GPW5, GPW6, GPW7, PH1,
                        PH1N, PH2, PH2N);

X9: ELSHFT14 PORT MAP ( S02020061, S02028035, S02028036, S02028037, S02028038,
                        S02028039, S02028040, S02028041, S02028042, S02028043,
                        S02028044, S02028045, S02028046, S02028047, S02028048,
                        PH1, PH1N, PH2, PH2N);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELPRO7 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELPRO7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:37:04
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELPRO7 is
     port(
PPINA: inout DotX;
PPINB: inout DotX;
PPINC: inout DotX;
PPIND: inout DotX;
PPINE: inout DotX;
PPINF: inout DotX;
PPING: inout DotX;
LOG0: out MVL7;
LOG1: out MVL7;
LOG2: out MVL7;
LOG3: out MVL7;
SDIN: in MVL7;
CSEL0: inout MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
PH2N: in MVL7;
PROGEN: in MVL7;
INJ1: in MVL7;
INJ2: in MVL7;
GDAT0: in MVL7;
GDAT1: in MVL7;
GDAT2: in MVL7;
GDAT3: in MVL7;
GDAT4: in MVL7;
GDAT5: in MVL7;
GDAT6: in MVL7;
GDAT7: in MVL7;
OUTPA: in MVL7;
OUTPB: in MVL7;
OUTPC: in MVL7;
OUTPD: in MVL7;
OUTPE: in MVL7;
OUTPF: in MVL7;
OUTPG: in MVL7;
INPINA: inout MVL7;
INPINB: inout MVL7;
INPINC: inout MVL7;
INPIND: inout MVL7;
INPINE: inout MVL7;
INPINF: inout MVL7;
INPING: inout MVL7 );
end ELPRO7;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELPRO7 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  S02050007 : MVL7;
    signal  OCLA : MVL7;
```

```
    signal  INVA : MVL7;
    signal  OENA : MVL7;
    signal  S02050016 : MVL7;
    signal  OCLB : MVL7;
    signal  INVB : MVL7;
    signal  OENB : MVL7;
    signal  S02050025 : MVL7;
    signal  OCLC : MVL7;
    signal  INVC : MVL7;
    signal  OENC : MVL7;
    signal  S02050034 : MVL7;
    signal  OCLD : MVL7;
    signal  INVD : MVL7;
    signal  OEND : MVL7;
    signal  S02050043 : MVL7;
    signal  OCLE : MVL7;
    signal  INVE : MVL7;
    signal  OENE : MVL7;
    signal  S02050052 : MVL7;
    signal  OCLF : MVL7;
    signal  INVF : MVL7;
    signal  OENF : MVL7;
    signal  S02050061 : MVL7;
    signal  OCLG : MVL7;
    signal  INVG : MVL7;
    signal  OENG : MVL7;
    signal  CSEL1 : MVL7;
    signal  CSEL2 : MVL7;
    signal  CSEL3 : MVL7;
    signal  CSEL4 : MVL7;
    signal  CSEL5 : MVL7;
    signal  CSEL6 : MVL7;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELPIN(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:36:47
--
component ELPIN     port(
PPIN: inout DotX;
INDAT: out MVL7;
INDATN: inout MVL7;
DPINOUT: in MVL7;
OPNCLCTR: in MVL7;
INVOUT: in MVL7;
OUTEN: in MVL7;
PROGEN: in MVL7);
end component;

-- COMPONENT GENERATED BY AVG FROM FILE ELSHIFT7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:36:51
--
component ELSHIFT7     port(
SDATIN: in MVL7;
SR0: inout MVL7;
SR1: inout MVL7;
SR2: inout MVL7;
SR3: inout MVL7;
SR4: inout MVL7;
SR5: inout MVL7;
SR6: inout MVL7;
PH2N: in MVL7;
PH1: in MVL7;
```

```
PH1N: in MVL7;
PH2: in MVL7);
end component;

-- COMPONENT GENERATED BY AVG FROM FILE ELFIX7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:36:53
--
component ELFIX7     port(
FX0: out MVL7;
FX1: out MVL7;
FX2: out MVL7;
FX3: out MVL7;
FX4: out MVL7;
FX5: out MVL7;
FX6: out MVL7;
INJ2: in MVL7;
EBLOCK: in MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
INJ1: in MVL7);
end component;

-- COMPONENT GENERATED BY AVG FROM FILE ELIN2(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:36:54
--
component ELIN2     port(
LIN0: in MVL7;
LIN1: in MVL7;
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
DOUT0: out MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
DOUT1: out MVL7;
EBLOCK0: in MVL7;
EBLOCK1: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7);
end component;


begin


PPADA: ELPIN PORT MAP ( PPINA, INPINA, S02050007, OUTPA, OCLA,
                        INVA, OENA, PROGEN);

PPADB: ELPIN PORT MAP ( PPINB, INPINB, S02050016, OUTPB, OCLB,
                        INVB, OENB, PROGEN);
```

```
PPADC: ELPIN PORT MAP ( PPINC, INPINC, S02050025, OUTPC, OCLC,
                        INVC, OENC, PROGEN);

PPADD: ELPIN PORT MAP ( PPIND, INPIND, S02050034, OUTPD, OCLD,
                        INVD, OEND, PROGEN);

PPADE: ELPIN PORT MAP ( PPINE, INPINE, S02050043, OUTPE, OCLE,
                        INVE, OENE, PROGEN);

PPADF: ELPIN PORT MAP ( PPINF, INPINF, S02050052, OUTPF, OCLF,
                        INVF, OENF, PROGEN);

PPADG: ELPIN PORT MAP ( PPING, INPING, S02050061, OUTPG, OCLG,
                        INVG, OENG, PROGEN);

X1: ELSHIFT7 PORT MAP ( SDIN, CSEL0, CSEL1, CSEL2, CSEL3,
                        CSEL4, CSEL5, CSEL6, PH2N, PH1,
                        PH1N, PH2);

X2: ELFIX7 PORT MAP ( INVA, INVB, INVC, INVD, INVE,
                      INVF, INVG, INJ2, CSEL2, GDAT1,
                      GDAT2, GDAT3, GDAT4, GDAT5, GDAT6,
                      GDAT7, INJ1);

X3: ELFIX7 PORT MAP ( OCLA, OCLB, OCLC, OCLD, OCLE,
                      OCLF, OCLG, INJ2, CSEL0, GDAT1,
                      GDAT2, GDAT3, GDAT4, GDAT5, GDAT6,
                      GDAT7, INJ1);

X4: ELFIX7 PORT MAP ( OENA, OENB, OENC, OEND, OENE,
                      OENF, OENG, INJ2, CSEL1, GDAT1,
                      GDAT2, GDAT3, GDAT4, GDAT5, GDAT6,
                      GDAT7, INJ1);

X5: ELIN2 PORT MAP ( INPINA, S02050007, INPINB, S02050016, INPINC,
                     S02050025, INPIND, S02050034, LOG0, GDAT0,
                     GDAT1, GDAT2, GDAT3, GDAT4, GDAT5,
                     GDAT6, GDAT7, LOG1, CSEL6, CSEL5,
                     INJ2, INJ1);

X6: ELIN2 PORT MAP ( INPIND, S02050034, INPINE, S02050043, INPINF,
                     S02050052, INPING, S02050061, LOG2, GDAT0,
                     GDAT1, GDAT2, GDAT3, GDAT4, GDAT5,
                     GDAT6, GDAT7, LOG3, CSEL4, CSEL3,
                     INJ2, INJ1);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELOUT14 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELOUT14(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:37:01
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELOUT14 is
     port(
LIN0: in MVL7;
LIN1: in MVL7;
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
DOUT0: out MVL7;
DOUT1: out MVL7;
DOUT2: out MVL7;
DOUT3: out MVL7;
DOUT4: out MVL7;
DOUT5: out MVL7;
DOUT6: out MVL7;
DOUT7: out MVL7;
DOUT8: out MVL7;
DOUT9: out MVL7;
DOUT10: out MVL7;
DOUT11: out MVL7;
DOUT12: out MVL7;
DOUT13: out MVL7;
EBLOCK0: in MVL7;
EBLOCK1: in MVL7;
EBLOCK2: in MVL7;
EBLOCK3: in MVL7;
EBLOCK4: in MVL7;
EBLOCK5: in MVL7;
EBLOCK6: in MVL7;
EBLOCK7: in MVL7;
EBLOCK8: in MVL7;
EBLOCK9: in MVL7;
EBLOCK10: in MVL7;
EBLOCK11: in MVL7;
EBLOCK12: in MVL7;
EBLOCK13: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7 );
end ELOUT14;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELOUT14 is

-- SIGNAL DECLARATIONS HERE
```

```
        signal GND : MVL7 :='0';
        signal VDD : MVL7 :='1';

--      END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELOUT7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--           Generated on 11-10-1991 at 16:36:56
--
component ELOUT7     port(
LIN0: in MVL7;
LIN1: in MVL7;
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
DOUT0: out MVL7;
DOUT1: out MVL7;
DOUT2: out MVL7;
DOUT3: out MVL7;
DOUT4: out MVL7;
DOUT5: out MVL7;
DOUT6: out MVL7;
EBLOCK0: in MVL7;
EBLOCK1: in MVL7;
EBLOCK2: in MVL7;
EBLOCK3: in MVL7;
EBLOCK4: in MVL7;
EBLOCK5: in MVL7;
EBLOCK6: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7);
end component;

begin

X1: ELOUT7 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                      LIN5, LIN6, LIN7, PDAT0, PDAT1,
                      PDAT2, PDAT3, PDAT4, PDAT5, PDAT6,
                      PDAT7, DOUT0, DOUT1, DOUT2, DOUT3,
                      DOUT4, DOUT5, DOUT6, EBLOCK0, EBLOCK1,
                      EBLOCK2, EBLOCK3, EBLOCK4, EBLOCK5, EBLOCK6,
                      INJ2, INJ1);

X2: ELOUT7 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                      LIN5, LIN6, LIN7, PDAT0, PDAT1,
                      PDAT2, PDAT3, PDAT4, PDAT5, PDAT6,
                      PDAT7, DOUT7, DOUT8, DOUT9, DOUT10,
                      DOUT11, DOUT12, DOUT13, EBLOCK7, EBLOCK8,
                      EBLOCK9, EBLOCK10, EBLOCK11, EBLOCK12, EBLOCK13,
                      INJ2, INJ1);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE
end structure ;
```

Subcell ELSHFT14 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELSHFT14(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:37:00
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELSHFT14 is
     port(
SDATIN: in MVL7;
SR0: inout MVL7;
SR1: inout MVL7;
SR2: inout MVL7;
SR3: inout MVL7;
SR4: inout MVL7;
SR5: inout MVL7;
SR6: inout MVL7;
SR7: inout MVL7;
SR8: inout MVL7;
SR9: inout MVL7;
SR10: inout MVL7;
SR11: inout MVL7;
SR12: inout MVL7;
SR13: inout MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
PH2N: in MVL7 );
end ELSHFT14;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELSHFT14 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELSHIFT7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--           Generated on 11-10-1991 at 16:36:51
--
component ELSHIFT7     port(
SDATIN: in MVL7;
SR0: inout MVL7;
SR1: inout MVL7;
SR2: inout MVL7;
SR3: inout MVL7;
SR4: inout MVL7;
SR5: inout MVL7;
SR6: inout MVL7;
PH2N: in MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7);
end component;
```

```
begin

X1: ELSHIFT7 PORT MAP ( SR7, SR0, SR1, SR2, SR3,
                        SR4, SR5, SR6, PH2N, PH1,
                        PH1N, PH2);

X2: ELSHIFT7 PORT MAP ( SDATIN, SR7, SR8, SR9, SR10,
                        SR11, SR12, SR13, PH2N, PH1,
                        PH1N, PH2);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELOUT7 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELOUT7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:56
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELOUT7 is
     port(
LIN0: in MVL7;
LIN1: in MVL7;
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
DOUT0: out MVL7;
DOUT1: out MVL7;
DOUT2: out MVL7;
DOUT3: out MVL7;
DOUT4: out MVL7;
DOUT5: out MVL7;
DOUT6: out MVL7;
EBLOCK0: in MVL7;
EBLOCK1: in MVL7;
EBLOCK2: in MVL7;
EBLOCK3: in MVL7;
EBLOCK4: in MVL7;
EBLOCK5: in MVL7;
EBLOCK6: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7 );
end ELOUT7;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELOUT7 is

-- SIGNAL DECLARATIONS HERE
        signal GND : MVL7 :='0';
        signal VDD : MVL7 :='1';

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELOG8(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--           Generated on 11-10-1991 at 16:36:49
--
component ELOG8     port(
LIN0: in MVL7;
LIN1: in MVL7;
```

```
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
COMBL: out MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
EBLOCK: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7);
end component;


begin


X1: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT0, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK0, INJ2, INJ1);

X2: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT1, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK1, INJ2, INJ1);

X3: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT2, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK2, INJ2, INJ1);

X4: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT3, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK3, INJ2, INJ1);

X5: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT4, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK4, INJ2, INJ1);

X6: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT5, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK5, INJ2, INJ1);

X7: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT6, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK6, INJ2, INJ1);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELIN2 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELIN2(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:54
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELIN2 is
     port(
LIN0: in MVL7;
LIN1: in MVL7;
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
DOUT0: out MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
DOUT1: out MVL7;
EBLOCK0: in MVL7;
EBLOCK1: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7 );
end ELIN2;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELIN2 is

-- SIGNAL DECLARATIONS HERE
        signal GND : MVL7 :='0';
        signal VDD : MVL7 :='1';

--      END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELOG8(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--           Generated on 11-10-1991 at 16:36:49
--
component ELOG8     port(
LIN0: in MVL7;
LIN1: in MVL7;
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
COMBL: out MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
```

```
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
EBLOCK: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7);
end component;


begin


X1: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT0, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK0, INJ2, INJ1);

X2: ELOG8 PORT MAP ( LIN0, LIN1, LIN2, LIN3, LIN4,
                     LIN5, LIN6, LIN7, DOUT1, PDAT0,
                     PDAT1, PDAT2, PDAT3, PDAT4, PDAT5,
                     PDAT6, PDAT7, EBLOCK1, INJ2, INJ1);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELFIX7 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELFIX7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:53
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELFIX7 is
     port(
FX0: out MVL7;
FX1: out MVL7;
FX2: out MVL7;
FX3: out MVL7;
FX4: out MVL7;
FX5: out MVL7;
FX6: out MVL7;
INJ2: in MVL7;
EBLOCK: in MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
INJ1: in MVL7 );
end ELFIX7;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELFIX7 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELFIX1(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--           Generated on 11-10-1991 at 16:36:47
--
component ELFIX1     port(
FIXOUT: out MVL7;
CONTROL1: in MVL7;
CONTROL2: in MVL7;
INJCTR2: in MVL7;
INJCTR1: in MVL7);
end component;


begin


X1: ELFIX1 PORT MAP ( FX0, EBLOCK, PDAT0, INJ2, INJ1);

X2: ELFIX1 PORT MAP ( FX1, EBLOCK, PDAT1, INJ2, INJ1);

X3: ELFIX1 PORT MAP ( FX2, EBLOCK, PDAT2, INJ2, INJ1);
```

Subcell ELFIX7 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELFIX7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:53
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELFIX7 is
     port(
FX0: out MVL7;
FX1: out MVL7;
FX2: out MVL7;
FX3: out MVL7;
FX4: out MVL7;
FX5: out MVL7;
FX6: out MVL7;
INJ2: in MVL7;
EBLOCK: in MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
INJ1: in MVL7 );
end ELFIX7;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELFIX7 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELFIX1(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:36:47
--
component ELFIX1     port(
FIXOUT: out MVL7;
CONTROL1: in MVL7;
CONTROL2: in MVL7;
INJCTR2: in MVL7;
INJCTR1: in MVL7);
end component;


begin


X1: ELFIX1 PORT MAP ( FX0, EBLOCK, PDAT0, INJ2, INJ1);

X2: ELFIX1 PORT MAP ( FX1, EBLOCK, PDAT1, INJ2, INJ1);

X3: ELFIX1 PORT MAP ( FX2, EBLOCK, PDAT2, INJ2, INJ1);
```

```
X4: ELFIX1 PORT MAP ( FX3, EBLOCK, PDAT3, INJ2, INJ1);
X5: ELFIX1 PORT MAP ( FX4, EBLOCK, PDAT4, INJ2, INJ1);
X6: ELFIX1 PORT MAP ( FX5, EBLOCK, PDAT5, INJ2, INJ1);
X7: ELFIX1 PORT MAP ( FX6, EBLOCK, PDAT6, INJ2, INJ1);
-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE
end structure ;
```

Subcell ELSHIFT8 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELSHIFT8(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:52
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELSHIFT8 is
     port(
SDATIN: in MVL7;
SR0: inout MVL7;
SR1: inout MVL7;
SR2: inout MVL7;
SR3: inout MVL7;
SR4: inout MVL7;
SR5: inout MVL7;
SR6: inout MVL7;
SR7: inout MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7;
PH2N: in MVL7 );
end ELSHIFT8;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELSHIFT8 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELDELAY(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--           Generated on 11-10-1991 at 16:36:45
--
component ELDELAY     port(
DATA: in MVL7;
DOUT: inout MVL7;
CK2N: in MVL7;
CK1: in MVL7;
CK1N: in MVL7;
CK2: in MVL7);
end component;


begin


X1: ELDELAY PORT MAP ( SDATIN, SR7, PH2N, PH1, PH1N,
                       PH2);

X2: ELDELAY PORT MAP ( SR7, SR6, PH2N, PH1, PH1N,
                       PH2);

X3: ELDELAY PORT MAP ( SR6, SR5, PH2N, PH1, PH1N,
                       PH2);
```

```
X4: ELDELAY PORT MAP ( SR5, SR4, PH2N, PH1, PH1N,
                       PH2);

X5: ELDELAY PORT MAP ( SR4, SR3, PH2N, PH1, PH1N,
                       PH2);

X6: ELDELAY PORT MAP ( SR3, SR2, PH2N, PH1, PH1N,
                       PH2);

X7: ELDELAY PORT MAP ( SR2, SR1, PH2N, PH1, PH1N,
                       PH2);

X8: ELDELAY PORT MAP ( SR1, SR0, PH2N, PH1, PH1N,
                       PH2);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELSHIFT7 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELSHIFT7(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:51
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELSHIFT7 is
     port(
SDATIN: in MVL7;
SR0: inout MVL7;
SR1: inout MVL7;
SR2: inout MVL7;
SR3: inout MVL7;
SR4: inout MVL7;
SR5: inout MVL7;
SR6: inout MVL7;
PH2N: in MVL7;
PH1: in MVL7;
PH1N: in MVL7;
PH2: in MVL7 );
end ELSHIFT7;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELSHIFT7 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELDELAY(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--            Generated on 11-10-1991 at 16:36:45
--
component ELDELAY     port(
DATA: in MVL7;
DOUT: inout MVL7;
CK2N: in MVL7;
CK1: in MVL7;
CK1N: in MVL7;
CK2: in MVL7);
end component;


begin


X1: ELDELAY PORT MAP ( SDATIN, SR6, PH2N, PH1, PH1N,
                       PH2);

X2: ELDELAY PORT MAP ( SR6, SR5, PH2N, PH1, PH1N,
                       PH2);

X3: ELDELAY PORT MAP ( SR5, SR4, PH2N, PH1, PH1N,
                       PH2);

X4: ELDELAY PORT MAP ( SR4, SR3, PH2N, PH1, PH1N,
```

```
                    PH2);

X5: ELDELAY PORT MAP ( SR3, SR2, PH2N, PH1, PH1N,
                    PH2);

X6: ELDELAY PORT MAP ( SR2, SR1, PH2N, PH1, PH1N,
                    PH2);

X7: ELDELAY PORT MAP ( SR1, SR0, PH2N, PH1, PH1N,
                    PH2);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELOG8 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELOG8(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:49
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELOG8 is
     port(
LIN0: in MVL7;
LIN1: in MVL7;
LIN2: in MVL7;
LIN3: in MVL7;
LIN4: in MVL7;
LIN5: in MVL7;
LIN6: in MVL7;
LIN7: in MVL7;
COMBL: out MVL7;
PDAT0: in MVL7;
PDAT1: in MVL7;
PDAT2: in MVL7;
PDAT3: in MVL7;
PDAT4: in MVL7;
PDAT5: in MVL7;
PDAT6: in MVL7;
PDAT7: in MVL7;
EBLOCK: in MVL7;
INJ2: in MVL7;
INJ1: in MVL7 );
end ELOG8;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELOG8 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  COMBLN : DotX;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

-- COMPONENT GENERATED BY AVG FROM FILE ELOG1(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--           Generated on 11-10-1991 at 16:36:46
--
component ELOG1     port(
LOGIN: in MVL7;
CONTROL1: in MVL7;
CONTROL2: in MVL7;
BITOUT: inout DotX;
INJCTR1: in MVL7;
INJCTR2: in MVL7);
end component;

component pchpu
-- P-CHANNEL MOS TRANSISTOR USED AS A PULL-UP RESISTOR
  port( PULLUP   : inout DotX);
end component;
component inverter
```

```
  port(input: in MVL7; output: out MVL7);
end component;

begin


X1: ELOG1 PORT MAP ( LIN0, EBLOCK, PDAT0, COMBLN, INJ1,
                     INJ2);

X10: PCHPU PORT MAP ( COMBLN);

X2: ELOG1 PORT MAP ( LIN1, EBLOCK, PDAT1, COMBLN, INJ1,
                     INJ2);

X3: ELOG1 PORT MAP ( LIN2, EBLOCK, PDAT2, COMBLN, INJ1,
                     INJ2);

X4: ELOG1 PORT MAP ( LIN3, EBLOCK, PDAT3, COMBLN, INJ1,
                     INJ2);

X5: ELOG1 PORT MAP ( LIN4, EBLOCK, PDAT4, COMBLN, INJ1,
                     INJ2);

X6: ELOG1 PORT MAP ( LIN5, EBLOCK, PDAT5, COMBLN, INJ1,
                     INJ2);

X7: ELOG1 PORT MAP ( LIN6, EBLOCK, PDAT6, COMBLN, INJ1,
                     INJ2);

X8: ELOG1 PORT MAP ( LIN7, EBLOCK, PDAT7, COMBLN, INJ1,
                     INJ2);

X9: INVERTER PORT MAP ( COMBLN, COMBL);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELPIN VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELPIN(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:47
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELPIN is
     port(
PPIN: inout DotX;
INDAT: out MVL7;
INDATN: inout MVL7;
DPINOUT: in MVL7;
OPNCLCTR: in MVL7;
INVOUT: in MVL7;
OUTEN: in MVL7;
PROGEN: in MVL7 );
end ELPIN;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELPIN is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  S01076019 : MVL7;
    signal  S01045014 : MVL7;
    signal  S01045021 : MVL7;
    signal  S01017039 : MVL7;
    signal  S01013027 : MVL7;
    signal  S01022025 : MVL7;
    signal  S01014015 : MVL7;
    signal  S01019061 : MVL7;
    signal  DPSEL : DotX;
    signal  S01028014 : MVL7;
    signal  S01021014 : MVL7;
    signal  S01013014 : MVL7;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

component tripad
  port(CHIP : out MVL7;
       OUTP : in MVL7;
       OUTN : in MVL7;
       PPIN  : inout DotX);
end component;
component inverter
  port(input: in MVL7; output: out MVL7);
end component;
component nand2
  port(ND2IN1: in MVL7; ND2IN2 : in MVL7; ND2OUT: out MVL7);
end component;
component tgate
  port(p1 : in MVL7;
       p2 : in MVL7;
       g : in MVL7;
       d  : inout DotX);
end component;
component nor2
```

```
  port(NR2IN1: in MVL7; NR2IN2 : in MVL7; NR2OUT: out MVL7);
end component;

begin

PAD1: TRIPAD PORT MAP ( S01076019, S01045014, S01045021, PPIN);

X1: INVERTER PORT MAP ( DPINOUT, S01017039);

X10: NAND2 PORT MAP ( OUTEN, S01013027, S01022025);

X11: INVERTER PORT MAP ( S01022025, S01014015);

X12: TGATE PORT MAP ( INVOUT, S01019061, S01017039, DPSEL);

X14: TGATE PORT MAP ( S01019061, INVOUT, DPINOUT, DPSEL);

X15: INVERTER PORT MAP ( INVOUT, S01019061);

X2: INVERTER PORT MAP ( S01076019, INDATN);

X3: INVERTER PORT MAP ( INDATN, INDAT);

X4: NAND2 PORT MAP ( DPSEL, S01028014, S01045014);

X5: NOR2 PORT MAP ( DPSEL, S01022025, S01045021);

X6: INVERTER PORT MAP ( S01021014, S01028014);

X7: NAND2 PORT MAP ( S01013014, S01014015, S01021014);

X8: INVERTER PORT MAP ( OPNCLCTR, S01013014);

X9: INVERTER PORT MAP ( PROGEN, S01013027);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELFIX1 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELFIX1(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:47
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELFIX1 is
     port(
FIXOUT: out MVL7;
CONTROL1: in MVL7;
CONTROL2: in MVL7;
INJCTR2: in MVL7;
INJCTR1: in MVL7 );
end ELFIX1;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELFIX1 is

-- SIGNAL DECLARATIONS HERE
        signal GND : MVL7 :='0';
        signal VDD : MVL7 :='1';
    signal  S01030039 : MVL7;
    signal  FIXNOT : DotX;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

component nand2
  port(ND2IN1: in MVL7; ND2IN2 : in MVL7; ND2OUT: out MVL7);
end component;
component EEPPM1
  port(DRAIN : inout DotX;
       GATE : in MVL7;
       INJ1 : in MVL7;
       INJ2 : in MVL7);
end component;
component inverter
  port(input: in MVL7; output: out MVL7);
end component;
component pchpu
-- P-CHANNEL MOS TRANSISTOR USED AS A PULL-UP RESISTOR
  port( PULLUP  : inout DotX);
end component;

begin


X1: NAND2 PORT MAP ( CONTROL1, CONTROL2, S01030039);

X2: EEPPM1 PORT MAP ( FIXNOT, S01030039, INJCTR1, INJCTR2);

X3: INVERTER PORT MAP ( FIXNOT, FIXOUT);

X4: PCHPU PORT MAP ( FIXNOT);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELOG1 VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELOG1(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:46
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELOG1 is
     port(
LOGIN: in MVL7;
CONTROL1: in MVL7;
CONTROL2: in MVL7;
BITOUT: inout DotX;
INJCTR1: in MVL7;
INJCTR2: in MVL7 );
end ELOG1;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELOG1 is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  S01030039 : MVL7;
    signal  S01035028 : MVL7;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

component nand2
  port(ND2IN1: in MVL7; ND2IN2 : in MVL7; ND2OUT: out MVL7);
end component;
component EEPPM1
  port(DRAIN : inout DotX;
       GATE : in MVL7;
       INJ1 : in MVL7;
       INJ2 : in MVL7);
end component;
component nchan3
  port(Gate: in MVL7;
       Drain: out DotX;
       Source: in MVL7);
end component;

begin


X1: NAND2 PORT MAP ( CONTROL1, CONTROL2, S01030039);

X2: EEPPM1 PORT MAP ( S01035028, S01030039, INJCTR1, INJCTR2);

X3: NCHAN3 PORT MAP ( LOGIN, BITOUT, S01035028);

-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell ELDELAY VHDL Model

```
-- VHDL GENERATED BY AVG FROM FILE ELDELAY(.NPL)
--      Automatic VHDL Generator (AVG) V 0.98 (Developmental) - 8/11/91
--      Adapted by Joe Breen,  AFIT ENS, WPAFB, OH, from
--      the Omation SCHEMA SPICE netlist generator.
--
--     Generated on 11-10-1991 at 16:36:45
--
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity ELDELAY is
     port(
DATA: in MVL7;
DOUT: inout MVL7;
CK2N: in MVL7;
CK1: in MVL7;
CK1N: in MVL7;
CK2: in MVL7 );
end ELDELAY;
-- END OF AUTOMATICALLY GENERATED VHDL entity

architecture structure of ELDELAY is

-- SIGNAL DECLARATIONS HERE
         signal GND : MVL7 :='0';
         signal VDD : MVL7 :='1';
    signal  CNODE1 : DotX;
    signal  STORE1N : MVL7;
    signal  CNODE2 : DotX;

--       END SIGNALS

-- COMPONENT DECLARATIONS HERE

component tgate
  port(p1 : in MVL7;
       p2 : in MVL7;
       g : in MVL7;
       d  : inout DotX);
end component;
component inverter
  port(input: in MVL7; output: out MVL7);
end component;
component clkdinv
  port(INPUT : in MVL7;
       OUTPUT  : inout DotX;
       PH1  : in MVL7;
       PH1N : in MVL7);
end component;

begin


X1: TGATE PORT MAP ( CK1, CK1N, DATA, CNODE1);

X2: INVERTER PORT MAP ( CNODE1, STORE1N);

X3: TGATE PORT MAP ( CK2, CK2N, STORE1N, CNODE2);

X4: INVERTER PORT MAP ( CNODE2, DOUT);

X5: CLKDINV PORT MAP ( STORE1N, CNODE1, CK1N, CK1);

X6: CLKDINV PORT MAP ( DOUT, CNODE2, CK2N, CK2);
```

```
-- END OF AUTOMATICALLY GENERATED VHDL STRUCTURE

end structure ;
```

Subcell TRIPAD VHDL Model

```
-----------------------------------------------------------------
-- Date:  02 Oct 91
-- Version:  1
--
-- Unix filename:  tripad.vhd
--
-- Function:  This file is a behavioral description of a tri-state
-- I/O pad.  The pad has inputs of OUTP and OUTN which are the gates
-- of a P-Channel and an N-Channel drive transistor respectively.
-- The output from the pad is CHIP which is a direct connection to
-- the metal pad.
-----------------------------------------------------------------
Library ZYCAD;
use ZYCAD.types.all;
use WORK.all;

entity tripad is
  port(CHIP : out MVL7;
       OUTP : in MVL7;
       OUTN : in MVL7;
       PPIN  : inout DotX);
end tripad;

architecture behavioral of tripad is
    Signal  node1 : MVL7;

begin
process

begin

    if ( MVL7toBIT(PPIN) = '1' ) then CHIP <= '1';
      elsif (MVL7toBIT(PPIN)='0' ) then CHIP <= '0';
    end if;

    if ( MVL7toBIT(OUTN) = '1' ) then PPIN <= '0';
      elsif (MVL7toBIT(OUTP)='0' ) then PPIN <= '1';
      else PPIN <= 'Z';
    end if;

    wait on OUTP, OUTN, PPIN;
end process;
end behavioral;
```

Appendix H: Detailed Programming Model for Microcircuit 2.

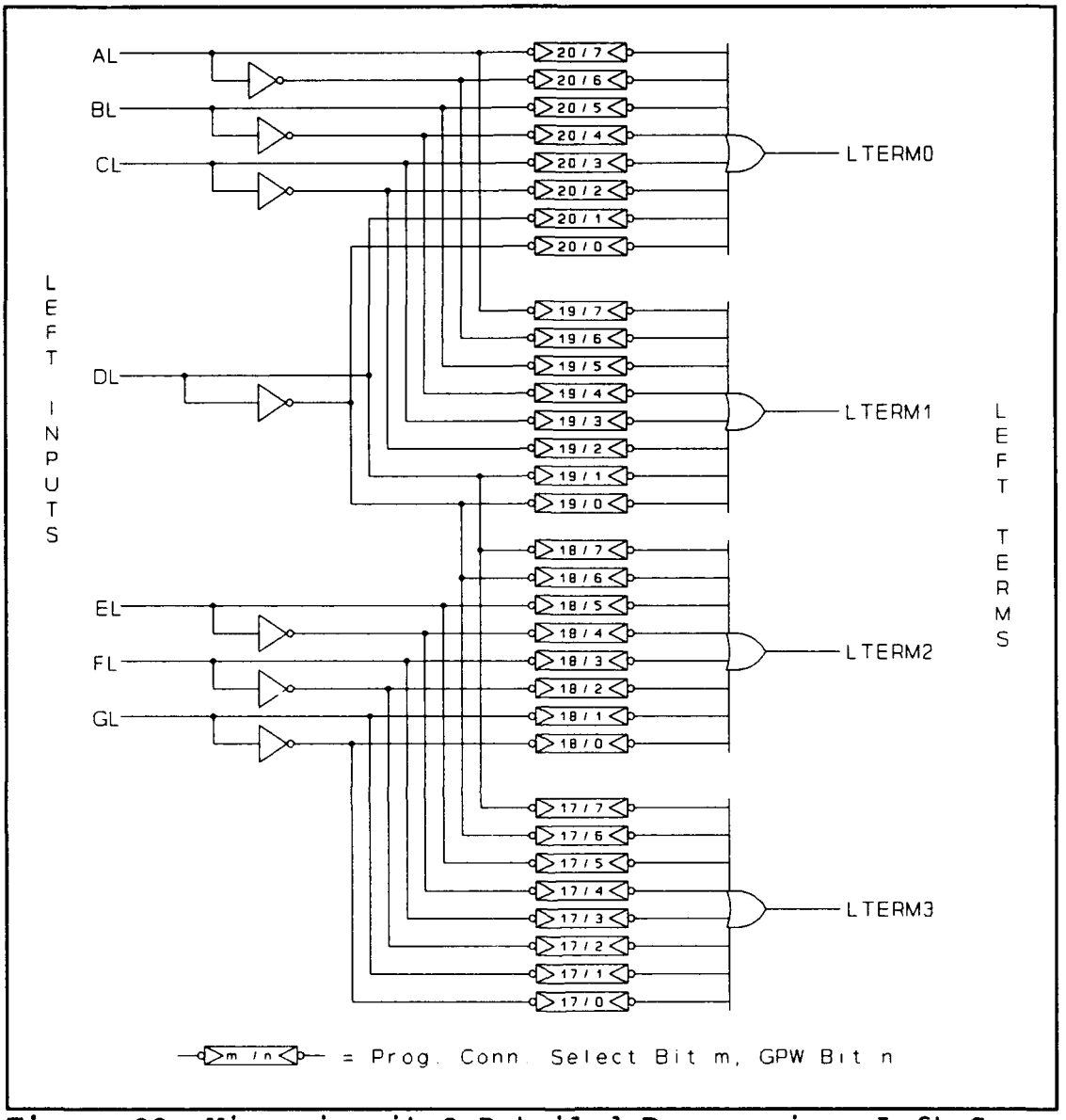


Figure 38. Microcircuit 2 Detailed Programming: Left Group Inputs.

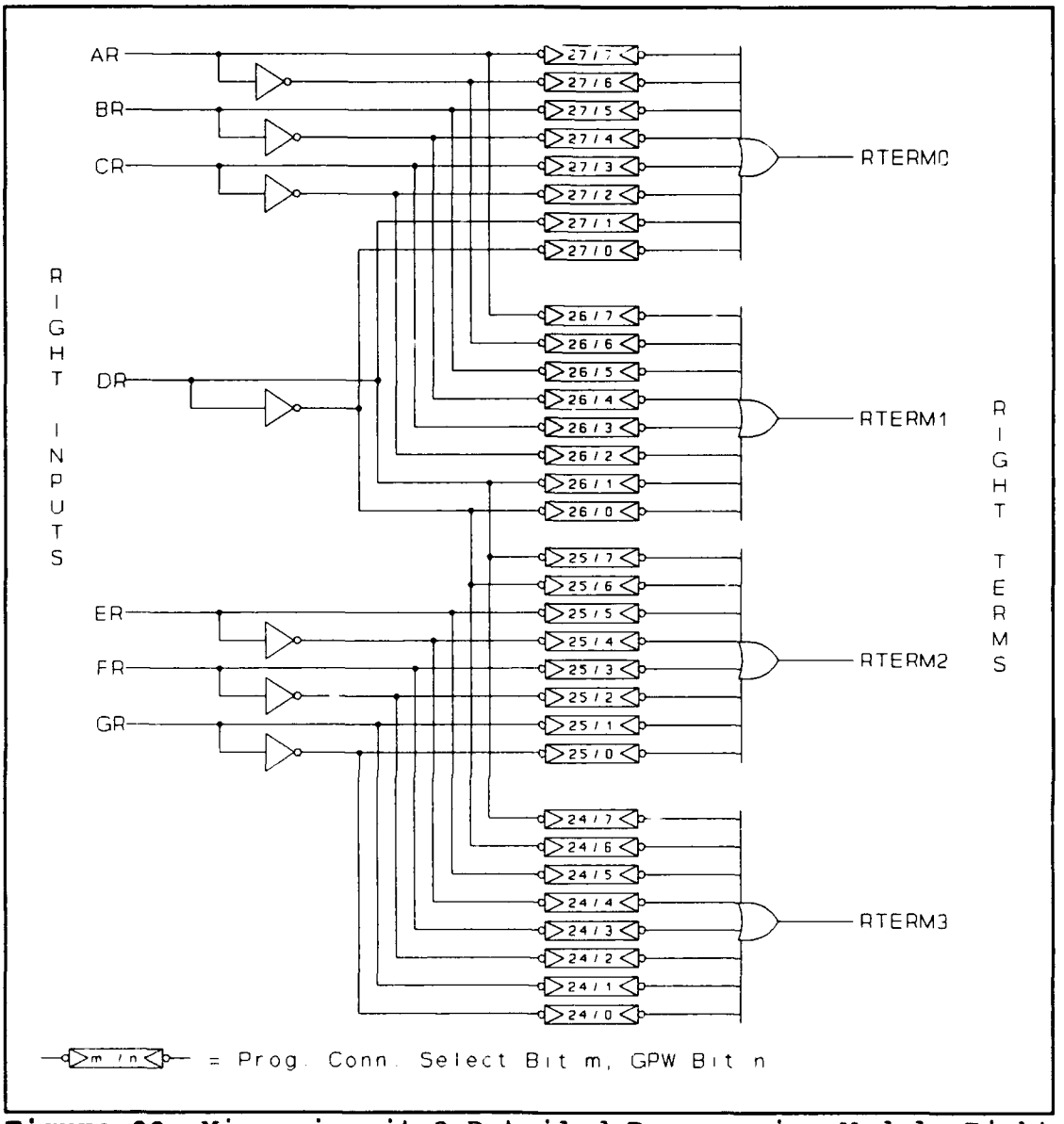


Figure 39. Microcircuit 2 Detailed Programming Model: Right Group Inputs.

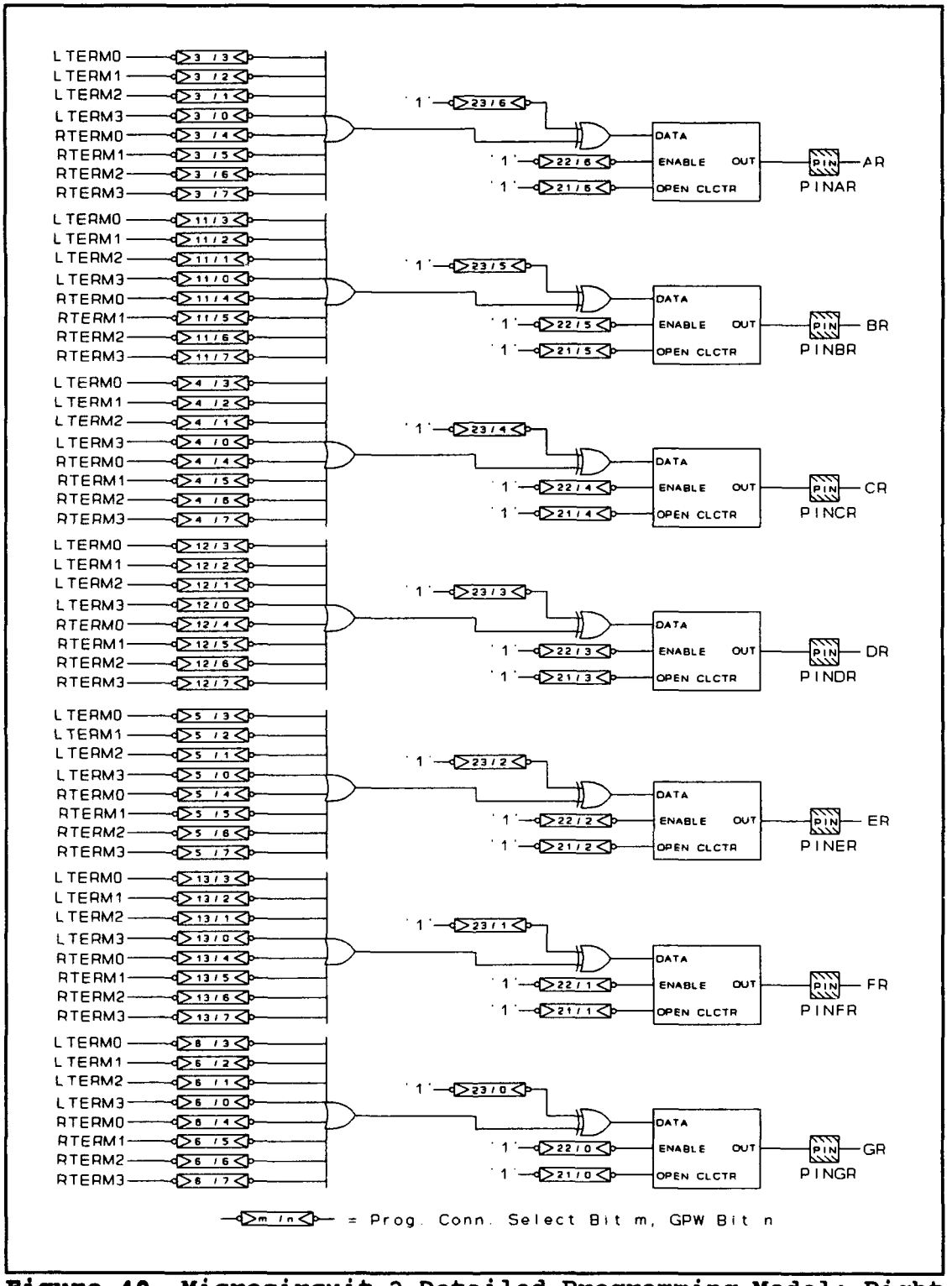


Figure 40. Microcircuit 2 Detailed Programming Model: Right Group Outputs.

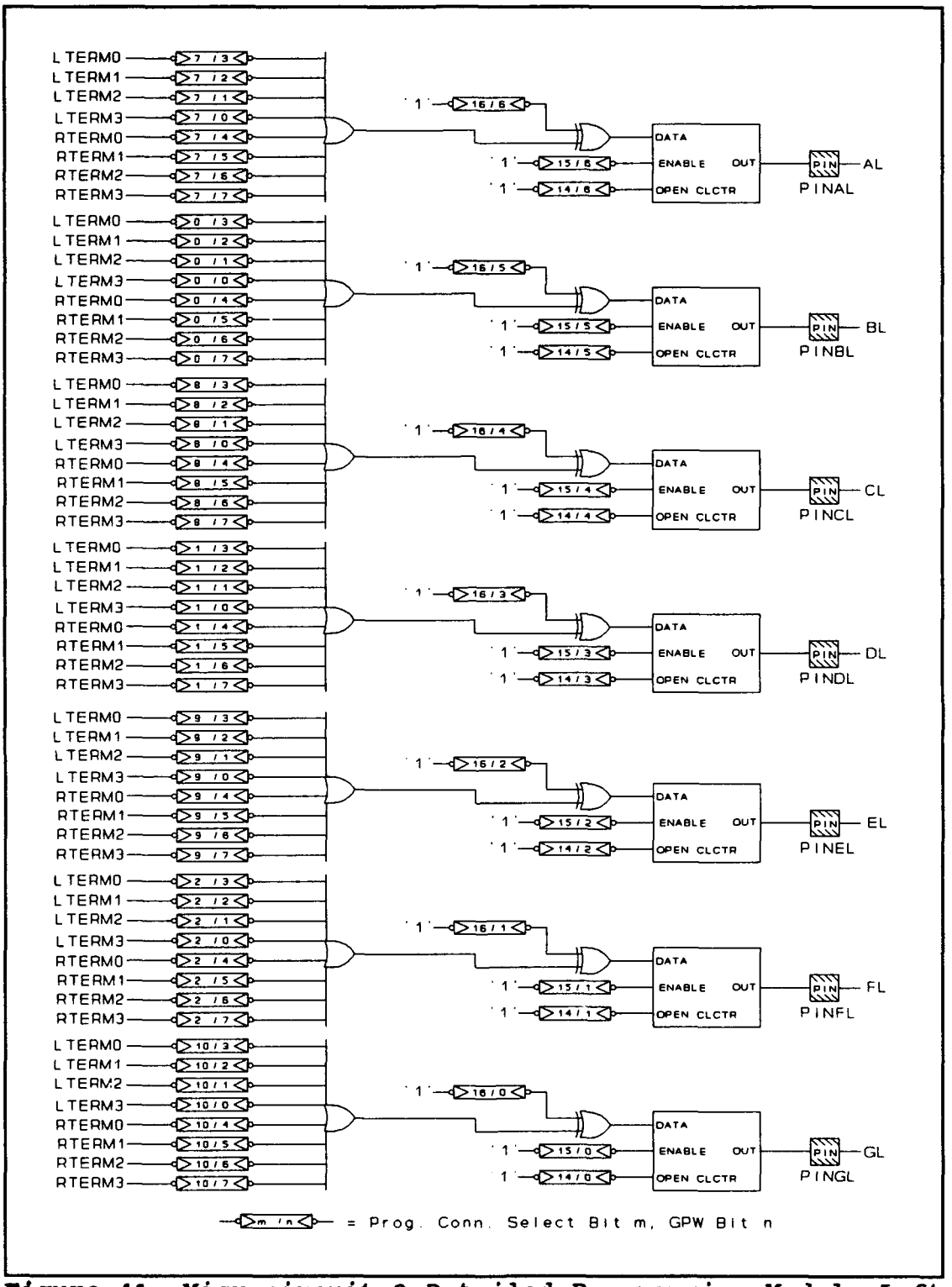


Figure 41. Microcircuit 2 Detailed Programming Model: Left Group Outputs.

Appendix I: Emulation Program Words

For Microcircuit 2.

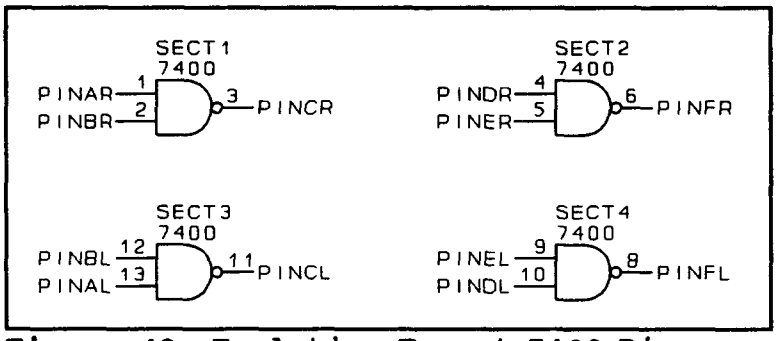


Figure 42. Emulation Target 7400 Diagram.

Table 9. Emulation Target 7400 Program Table.

Program Sheet for: 7400

LABEL	DESCRIPTION	GPW7 SRB35	GPW6 34	GPW5 33	GPW4 32	GPW3 31	GPW2 30	GPW1 29	GPW0 28	GPW	-> SEL
LGOUPBL	Out: LEFT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	0
LGOUPDL	Out: LEFT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	1
LGOUPFL	Out: LEFT F	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 1	LT3= 0	2	-> 2
LGOUPGR	Out: RIGHT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	6
LGOUPER	Out: RIGHT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	5
LGOUPCR	Out: RIGHT C	RT3= 0	RT2= 0	RT1= 0	RT0= 1	LT0= 0	LT1= 0	LT2= 0	LT3= 0	16	-> 4
LGOUPAR	Out: RIGHT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	3
LGOUPAL	Out: LEFT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	7
LGOUPCL	Out: LEFT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 1	LT1= 0	LT2= 0	LT3= 0	8	-> 8
LGOUPEL	Out: LEFT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	9
LGOUPGL	Out: LEFT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	10
LGOUPFR	Out: RIGHT F	RT3= 0	RT2= 1	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	64	-> 13
LGOUPDR	Out: RIGHT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	12
LGOUPBR	Out:RIGHT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	11
X3/CSEL0	OC: LEFT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	14
X3/CSEL1	OEN: LEFT Pins	n/a	0 OENA	0 OENB	0 OENC	1 OEND	0 OENE	0 OENF	1 OENG	18	-> 15
X3/CSEL2	Inv: LEFT Pins	n/a	0 INVA	0 INVB	0 INVC	0 INVD	0 INVE	0 INVF	0 INVG	0	16
X3/CSEL3	LTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	17
X3/CSEL4	LTERM2 Program	D	0 D'	1 E	0 E'	1 F	0 F'	0 G	0 G'	80	-> 18
X3/CSEL5	LTERM1 Program	A	0 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	0	19
X3/CSEL6	LTERM0 Program	A	0 A'	1 B	0 B'	1 C	0 C'	0 D	0 D'	80	-> 20
X5/CSEL0	OC:RIGHT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	21
X5/CSEL1	OEN:RIGHT Pins	n/a	0 OENA	0 OENB	0 OENC	1 OEND	0 OENE	0 OENF	1 OENG	18	-> 22
X5/CSEL2	Inv:RIGHT Pins	n/a	0 INVA	0 INVB	0 INVC	0 INVD	0 INVE	0 INVF	0 INVG	0	23
X5/CSEL3	RTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	24
X5/CSEL4	RTERM2 Program	D	0 D'	1 E	0 E'	1 F	0 F'	0 G	0 G'	80	-> 25
X5/CSEL5	RTERM1 Program	A	0 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	0	26
X5/CSEL6	RTERM0 Program	A	0 A'	1 B	0 B'	1 C	0 C'	0 D	0 D'	80	-> 27

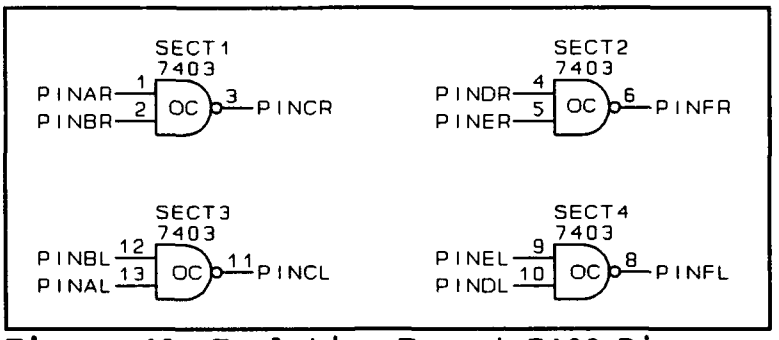


Figure 43. Emulation Target 7403 Diagram.

Table 10. Emulation Target 7403 Program Table.

Program Sheet for: 7403

LABEL	DESCRIPTION	GPW7 SRB35	GPW6 34	GPW5 33	GPW4 32	GPW3 31	GPW2 30	GPW1 29	GPW0 28	GPW	-> SEL
LGOUPBL	Out: LEFT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	0
LGOUPDL	Out: LEFT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	1
LGOUPFL	Out: LEFT F	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 1	LT3= 0	2	-> 2
LGOUPGR	Out: RIGHT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	6
LGOUPER	Out: RIGHT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	5
LGOUPCR	Out: RIGHT C	RT3= 0	RT2= 0	RT1= 0	RT0= 1	LT0= 0	LT1= 0	LT2= 0	LT3= 0	16	-> 4
LGOUPAR	Out: RIGHT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	3
LGOUPAL	Out: LEFT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	7
LGOUPCL	Out: LEFT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 1	LT1= 0	LT2= 0	LT3= 0	8	-> 8
LGOUPEL	Out: LEFT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	9
LGOUPGL	Out: LEFT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	10
LGOUPFR	Out: RIGHT F	RT3= 0	RT2= 1	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	64	-> 13
LGOUPDR	Out: RIGHT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	12
LGOUPBR	Out:RIGHT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	11
X3/CSEL0	OC: LEFT Pins	n/a 0	OCLA 0	OCLB 0	OCLC 1	OCLD 0	OCLE 0	OCLF 1	OCLG 0	18	-> 14
X3/CSEL1	OEN: LEFT Pins	n/a 0	OENA 0	OENB 0	OENC 1	OEND 0	OENE 0	OENF 1	OENG 0	18	-> 15
X3/CSEL2	Inv: LEFT Pins	n/a 0	INVA 0	INVB 0	INVC 0	INVD 0	INVE 0	INVF 0	INVG 0	0	16
X3/CSEL3	LTERM3 Program	D 0	D' 0	E 0	E' 0	F 0	F' 0	G 0	G' 0	0	17
X3/CSEL4	LTERM2 Program	D 0	D' 1	E 0	E' 1	F 0	F' 0	G 0	G' 0	80	-> 18
X3/CSEL5	LTERM1 Program	A 0	A' 0	B 0	B' 0	C 0	C' 0	D 0	D' 0	0	19
X3/CSEL6	LTERM0 Program	A 0	A' 1	B 0	B' 1	C 0	C' 0	D 0	D' 0	80	-> 20
X5/CSEL0	OC:RIGHT Pins	n/a 0	OCLA 0	OCLB 0	OCLC 1	OCLD 0	OCLE 0	OCLF 1	OCLG 0	18	-> 21
X5/CSEL1	OEN:RIGHT Pins	n/a 0	OENA 0	OENB 0	OENC 1	OEND 0	OENE 0	OENF 1	OENG 0	18	-> 22
X5/CSEL2	Inv:RIGHT Pins	n/a 0	INVA 0	INVB 0	INVC 0	INVD 0	INVE 0	INVF 0	INVG 0	0	23
X5/CSEL3	RTERM3 Program	D 0	D' 0	E 0	E' 0	F 0	F' 0	G 0	G' 0	0	24
X5/CSEL4	RTERM2 Program	D 0	D' 1	E 0	E' 1	F 0	F' 0	G 0	G' 0	80	-> 25
X5/CSEL5	RTERM1 Program	A 0	A' 0	B 0	B' 0	C 0	C' 0	D 0	D' 0	0	26
X5/CSEL6	RTERM0 Program	A 0	A' 1	B 0	B' 1	C 0	C' 0	D 0	D' 0	80	-> 27

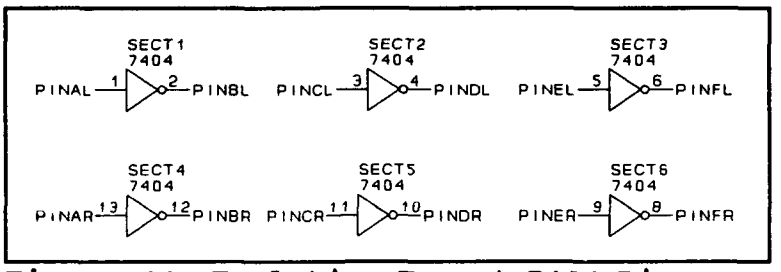


Figure 44. Emulation Target 7404 Diagram.

Table 11. Emulation Target 7404 Program Table.

Program Sheet for: 7404

LABEL	DESCRIPTION	GPW7 SRB35	GPW6 34	GPW5 33	GPW4 32	GPW3 31	GPW2 30	GPW1 29	GPW0 28	GPW	-> SEL
LGOUPBL	Out: LEFT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 1	LT1= 0	LT2= 0	LT3= 0	8	-> 0
LGOUPDL	Out: LEFT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 1	LT2= 0	LT3= 0	4	-> 1
LGOUPFL	Out: LEFT F	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 1	LT3= 0	2	-> 2
LGOUPGR	Out: RIGHT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	6
LGOUPER	Out: RIGHT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	5
LGOUPCR	Out: RIGHT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	4
LGOUPAR	Out: RIGHT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	3
LGOUPAL	Out: LEFT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	7
LGOUPCL	Out: LEFT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	8
LGOUPEL	Out: LEFT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	9
LGOUPGL	Out: LEFT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	10
LGOUPFR	Out: RIGHT F	RT3= 0	RT2= 1	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	64	-> 13
LGOUPDR	Out: RIGHT D	RT3= 0	RT2= 0	RT1= 1	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	32	-> 12
LGOUPBR	Out:RIGHT B	RT3= 0	RT2= 0	RT1= 0	RT0= 1	LT0= 0	LT1= 0	LT2= 0	LT3= 0	16	-> 11
X3/CSEL0	OC: LEFT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	14
X3/CSEL1	OEN: LEFT Pins	n/a	0 OENA	0 OENB	1 OENC	0 OEND	1 OENE	0 OENF	1 OENG	42	-> 15
X3/CSEL2	Inv: LEFT Pins	n/a	0 INVA	0 INVB	1 INVC	0 INVD	1 INVE	0 INVF	1 INVG	42	-> 16
X3/CSEL3	LTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	17
X3/CSEL4	LTERM2 Program	D	0 D'	0 E	1 E'	0 F	0 F'	0 G	0 G'	32	-> 18
X3/CSEL5	LTERM1 Program	A	0 A'	0 B	0 B'	0 C	1 C'	0 D	0 D'	8	-> 19
X3/CSEL6	LTERM0 Program	A	1 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	128	-> 20
X5/CSEL0	OC:RIGHT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	21
X5/CSEL1	OEN:RIGHT Pins	n/a	0 OENA	0 OENB	1 OENC	0 OEND	1 OENE	0 OENF	1 OENG	42	-> 22
X5/CSEL2	Inv:RIGHT Pins	n/a	0 INVA	0 INVB	1 INVC	0 INVD	1 INVE	0 INVF	1 INVG	42	-> 23
X5/CSEL3	RTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	24
X5/CSEL4	RTERM2 Program	D	0 D'	0 E	1 E'	0 F	0 F'	0 G	0 G'	32	-> 25
X5/CSEL5	RTERM1 Program	A	0 A'	0 B	0 B'	0 C	1 C'	0 D	0 D'	8	-> 26
X5/CSEL6	RTERM0 Program	A	1 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	128	-> 27

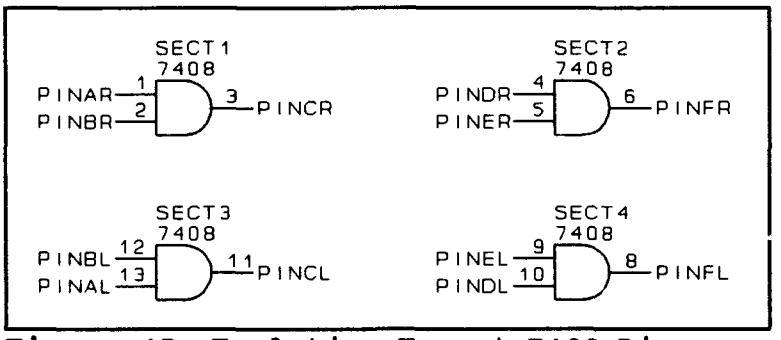


Figure 45. Emulation Target 7408 Diagram.

Table 12. Emulation Target 7408 Program Table.

Program Sheet for: 7408

LABEL	DESCRIPTION	GPW7 SRB35	GPW6 34	GPW5 33	GPW4 32	GPW3 31	GPW2 30	GPW1 29	GPW0 28	GPW	-> SEL
LGOUPBL	Out: LEFT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	0
LGOUPDL	Out: LEFT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	1
LGOUPFL	Out: LEFT F	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 1	LT3= 0	2	-> 2
LGOUPGR	Out: RIGHT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	6
LGOUPER	Out: RIGHT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	5
LGOUPCR	Out: RIGHT C	RT3= 0	RT2= 0	RT1= 0	RT0= 1	LT0= 0	LT1= 0	LT2= 0	LT3= 0	16	-> 4
LGOUPAR	Out: RIGHT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	3
LGOUPAL	Out: LEFT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	7
LGOUPCL	Out: LEFT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 1	LT1= 0	LT2= 0	LT3= 0	8	-> 8
LGOUPEL	Out: LEFT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	9
LGOUPGL	Out: LEFT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	10
LGOUPFR	Out: RIGHT F	RT3= 0	RT2= 1	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	64	-> 13
LGOUPDR	Out: RIGHT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	12
LGOUPBR	Out:RIGHT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	11
X3/CSEL0	OC: LEFT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	14
X3/CSEL1	OEN: LEFT Pins	n/a	0 OENA	0 OENB	0 OENC	1 OEND	0 OENE	0 OENF	1 OENG	18	-> 15
X3/CSEL2	Inv: LEFT Pins	n/a	0 INVA	0 INVB	0 INVC	1 INVD	0 INVE	0 INVF	1 INVG	18	-> 16
X3/CSEL3	LTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	17
X3/CSEL4	LTERM2 Program	D	0 D'	1 E	0 E'	1 F	0 F'	0 G	0 G'	80	-> 18
X3/CSEL5	LTERM1 Program	A	0 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	0	19
X3/CSEL6	LTERM0 Program	A	0 A'	1 B	0 B'	1 C	0 C'	0 D	0 D'	80	-> 20
X5/CSEL0	OC:RIGHT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	21
X5/CSEL1	OEN:RIGHT Pins	n/a	0 OENA	0 OENB	0 OENC	1 OEND	0 OENE	0 OENF	1 OENG	18	-> 22
X5/CSEL2	Inv:RIGHT Pins	n/a	0 INVA	0 INVB	0 INVC	1 INVD	0 INVE	0 INVF	1 INVG	18	-> 23
X5/CSEL3	RTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	24
X5/CSEL4	RTERM2 Program	D	0 D'	1 E	0 E'	1 F	0 F'	0 G	0 G'	80	-> 25
X5/CSEL5	RTERM1 Program	A	0 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	0	26
X5/CSEL6	RTERM0 Program	A	0 A'	1 B	0 B'	1 C	0 C'	0 D	0 D'	80	-> 27

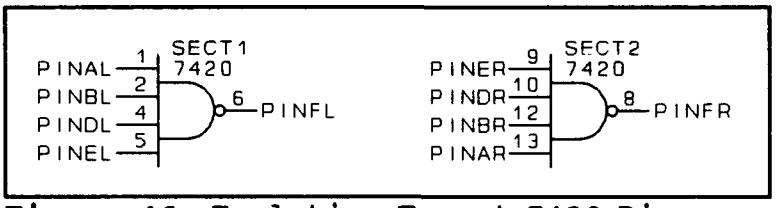


Figure 46. Emulation Target 7420 Diagram.

Table 13. Emulation Target 7420 Program Table.

Program Sheet for: 7420

LABEL	DESCRIPTION	GPW7 SRB35	GPW6 34	GPW5 33	GPW4 32	GPW3 31	GPW2 30	GPW1 29	GPW0 28	GPW	-> SEL
LGOUPBL	Out: LEFT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	0
LGOUPDL	Out: LEFT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	1
LGOUPFL	Out: LEFT F	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 1	LT1= 0	LT2= 1	LT3= 0	10	-> 2
LGOUPGR	Out: RIGHT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	6
LGOUPER	Out: RIGHT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	5
LGOUPCR	Out: RIGHT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	4
LGOUPAR	Out: RIGHT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	3
LGOUPAL	Out: LEFT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	7
LGOUPCL	Out: LEFT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	8
LGOUPEL	Out: LEFT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	9
LGOUPGL	Out: LEFT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	10
LGOUPFR	Out: RIGHT F	RT3= 0	RT2= 1	RT1= 0	RT0= 1	LT0= 0	LT1= 0	LT2= 0	LT3= 0	80	-> 13
LGOUPDR	Out: RIGHT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	12
LGOUPBR	Out:RIGHT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	11
X3/CSEL0	OC: LEFT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	14
X3/CSEL1	OEN: LEFT Pins	n/a	0 OENA	0 OENB	0 OENC	0 OEND	0 OENE	0 OENF	1 OENG	2	-> 15
X3/CSEL2	Inv: LEFT Pins	n/a	0 INVA	0 INVB	0 INVC	0 INVD	0 INVE	0 INVF	0 INVG	0	16
X3/CSEL3	LTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	17
X3/CSEL4	LTERM2 Program	D	0 D'	1 E	0 E'	1 F	0 F'	0 G	0 G'	80	-> 18
X3/CSEL5	LTERM1 Program	A	0 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	0	19
X3/CSEL6	LTERM0 Program	A	0 A'	1 B	0 B'	1 C	0 C'	0 D	0 D'	80	-> 20
X5/CSEL0	OC:RIGHT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	21
X5/CSEL1	OEN:RIGHT Pins	n/a	0 OENA	0 OENB	0 OENC	0 OEND	0 OENE	0 OENF	1 OENG	2	-> 22
X5/CSEL2	Inv:RIGHT Pins	n/a	0 INVA	0 INVB	0 INVC	0 INVD	0 INVE	0 INVF	0 INVG	0	23
X5/CSEL3	RTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	24
X5/CSEL4	RTERM2 Program	D	0 D'	1 E	0 E'	1 F	0 F'	0 G	0 G'	80	-> 25
X5/CSEL5	RTERM1 Program	A	0 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	0	26
X5/CSEL6	RTERM0 Program	A	0 A'	1 B	0 B'	1 C	0 C'	0 D	0 D'	80	-> 27

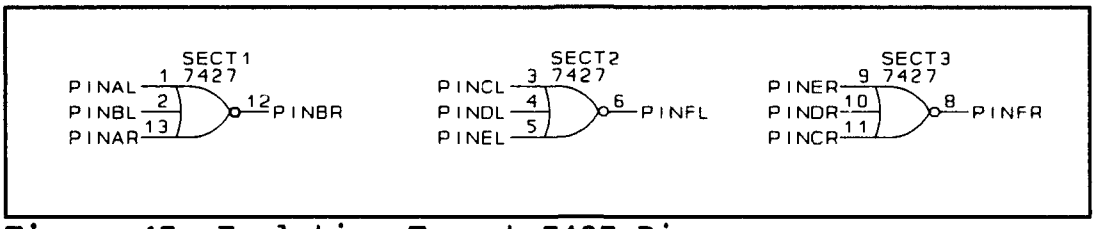


Figure 47. Emulation Target 7427 Diagram.

Table 14. Emulation Target 7427 Program Table.

Program Sheet for:7427

LABEL	DESCRIPTION	GPW7 SRB35	GPW6 34	GPW5 33	GPW4 32	GPW3 31	GPW2 30	GPW1 29	GPW0 28	GPW	-> SEL
LGOUPBL	Out: LEFT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	0
LGOUPDL	Out: LEFT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	1
LGOUPFL	Out: LEFT F	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 1	LT2= 1	LT3= 0	6	-> 2
LGOUPGR	Out: RIGHT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	6
LGOUPER	Out: RIGHT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	5
LGOUPCR	Out: RIGHT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	4
LGOUPAR	Out: RIGHT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	3
LGOUPAL	Out: LEFT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	7
LGOUPCL	Out: LEFT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	8
LGOUPEL	Out: LEFT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	9
LGOUPGL	Out: LEFT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	10
LGOUPFR	Out: RIGHT F	RT3= 0	RT2= 1	RT1= 1	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	96	-> 13
LGOUPDR	Out: RIGHT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	12
LGOUPBR	Out:RIGHT B	RT3= 0	RT2= 0	RT1= 0	RT0= 1	LT0= 1	LT1= 0	LT2= 0	LT3= 0	24	-> 11
X3/CSEL0	OC: LEFT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	14
X3/CSEL1	OEN: LEFT Pins	n/a	0 OENA	0 OENB	0 OENC	0 OEND	0 OENE	0 OENF	1 OENG	2	-> 15
X3/CSEL2	Inv: LEFT Pins	n/a	0 INVA	0 INVB	0 INVC	0 INVD	0 INVE	0 INVF	1 INVG	2	-> 16
X3/CSEL3	LTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	17
X3/CSEL4	LTERM2 Program	D	0 D'	0 E	1 E'	0 F	0 F'	0 G	0 G'	32	-> 18
X3/CSEL5	LTERM1 Program	A	0 A'	0 B	0 B'	0 C	1 C'	0 D	1 D'	10	-> 19
X3/CSEL6	LTERM0 Program	A	1 A'	0 B	1 B'	0 C	0 C'	0 D	0 D'	160	-> 20
X5/CSEL0	OC:RIGHT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	21
X5/CSEL1	OEN:RIGHT Pins	n/a	0 OENA	0 OENB	1 OENC	0 OEND	0 OENE	0 OENF	1 OENG	34	-> 22
X5/CSEL2	Inv:RIGHT Pins	n/a	0 INVA	0 INVB	1 INVC	0 INVD	0 INVE	0 INVF	1 INVG	34	-> 23
X5/CSEL3	RTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	24
X5/CSEL4	RTERM2 Program	D	0 D'	0 E	1 E'	0 F	0 F'	0 G	0 G'	32	-> 25
X5/CSEL5	RTERM1 Program	A	0 A'	0 B	0 B'	0 C	1 C'	0 D	1 D'	10	-> 26
X5/CSEL6	RTERM0 Program	A	1 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	128	-> 27

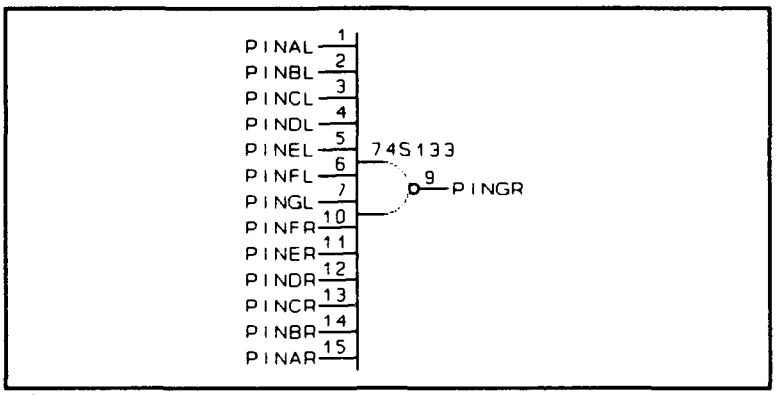


Figure 48. Emulation Target 74S133 Diagram.

Table 15. Emulation Target 74S133 Program Table.

Program Sheet for: 74S133

LABEL	DESCRIPTION	GPW7 SRB35	GPW6 34	GPW5 33	GPW4 32	GPW3 31	GPW2 30	GPW1 29	GPW0 28	GPW	-> SEL
LGOUPBL	Out: LEFT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	0
LGOUPDL	Out: LEFT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	1
LGOUPFL	Out: LEFT F	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	2
LGOUPGR	Out: RIGHT G	RT3= 0	RT2= 1	RT1= 0	RT0= 1	LT0= 1	LT1= 0	LT2= 1	LT3= 0	90	-> 6
LGOUPER	Out: RIGHT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	5
LGOUPCR	Out: RIGHT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	4
LGOUPAR	Out: RIGHT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	3
LGOUPAL	Out: LEFT A	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	7
LGOUPCL	Out: LEFT C	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	8
LGOUPEL	Out: LEFT E	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	9
LGOUPGL	Out: LEFT G	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	10
LGOUPFR	Out: RIGHT F	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	13
LGOUPDR	Out: RIGHT D	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	12
LGOUPBR	Out:RIGHT B	RT3= 0	RT2= 0	RT1= 0	RT0= 0	LT0= 0	LT1= 0	LT2= 0	LT3= 0	0	11
X3/CSEL0	OC: LEFT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	14
X3/CSEL1	OEN: LEFT Pins	n/a	0 OENA	0 OENB	0 OENC	0 OEND	0 OENE	0 OENF	0 OENG	0	15
X3/CSEL2	Inv: LEFT Pins	n/a	0 INVA	0 INVB	0 INVC	0 INVD	0 INVE	0 INVF	0 INVG	0	16
X3/CSEL3	LTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	17
X3/CSEL4	LTERM2 Program	D	0 D'	1 E	0 E'	1 F	0 F'	1 G	0 G'	1 85	-> 18
X3/CSEL5	LTERM1 Program	A	0 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	0	19
X3/CSEL6	LTERM0 Program	A	0 A'	1 B	0 B'	1 C	0 C'	1 D	0 D'	1 85	-> 20
X5/CSEL0	OC:RIGHT Pins	n/a	0 OCLA	0 OCLB	0 OCLC	0 OCLD	0 OCLE	0 OCLF	0 OCLG	0	21
X5/CSEL1	OEN:RIGHT Pins	n/a	0 OENA	0 OENB	0 OENC	0 OEND	0 OENE	0 OENF	0 OENG	1	1 -> 22
X5/CSEL2	Inv:RIGHT Pins	n/a	0 INVA	0 INVB	0 INVC	0 INVD	0 INVE	0 INVF	0 INVG	0	23
X5/CSEL3	RTERM3 Program	D	0 D'	0 E	0 E'	0 F	0 F'	0 G	0 G'	0	24
X5/CSEL4	RTERM2 Program	D	0 D'	1 E	0 E'	1 F	0 F'	1 G	0 G'	0 84	-> 25
X5/CSEL5	RTERM1 Program	A	0 A'	0 B	0 B'	0 C	0 C'	0 D	0 D'	0	26
X5/CSEL6	RTERM0 Program	A	0 A'	1 B	0 B'	1 C	0 C'	1 D	0 D'	1 85	-> 27

Appendix J: Results of VHDL Tests

For Microcircuit 2.

Table 16. VHDL Test Results for Emulation Target 7400.

INPUT		INPUT		INPUT		INPUT		OUTPUTS			
BL	AL	EL	DL	BR	AR	ER	DR	CL	FL	CR	FR
0	0	0	0	0	0	0	0	1	1	1	1
0	1	0	1	0	1	0	1	1	1	1	1
1	0	1	0	1	0	1	0	1	1	1	1
1	1	1	1	1	1	1	1	0	0	0	0

Table 17. VHDL Test Results for Emulation Target 7403.

INPUT		INPUT		INPUT		INPUT		OUTPUTS			
BL	AL	EL	DL	BR	AR	ER	DR	CL	FL	CR	FR
0	0	0	0	0	0	0	0	Z	Z	Z	Z
0	1	0	1	0	1	0	1	Z	Z	Z	Z
1	0	1	0	1	0	1	0	Z	Z	Z	Z
1	1	1	1	1	1	1	1	0	0	0	0

Table 18. VHDL Test Results for Emulation Target 7404.

IN	OUT	IN	OUT	IN	OUT	IN	OUT	IN	OUT	IN	OUT
ER	FR	CR	BR	AR	BR	EL	FL	CL	DL	AL	BL
0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	1	0	1	0	1	0	1	1	0
0	1	0	1	0	1	0	1	1	0	1	0
0	1	0	1	0	1	1	0	1	0	1	0
0	1	0	1	1	0	1	0	1	0	1	0
0	1	1	0	1	0	1	0	1	0	1	0
1	0	1	0	1	0	1	0	1	0	1	0

Table 19. VHDL Test Results for Emulation Target 7408.

INPUT		INPUT		INPUT		INPUT		OUTPUTS			
BL	AL	EL	DL	BR	AR	ER	DR	CL	FL	CR	FR
0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	0	0	0	0
1	0	1	0	1	0	1	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1

Table 20. VHDL Test Results for Emulation Target 7420.

INPUT				INPUT				OUTPUTS	
EL	DL	BL	AL	ER	DR	BR	AR	FL	FR
0	0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	0	1	1	1
0	0	1	0	0	0	1	0	1	1
0	0	1	1	0	0	1	1	1	1
0	1	0	0	0	1	0	0	1	1
0	1	0	1	0	1	0	1	1	1
0	1	1	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1
1	0	0	0	1	0	0	0	1	1
1	0	0	1	1	0	0	1	1	1
1	0	1	0	1	0	1	0	1	1
1	0	1	1	1	0	1	1	1	1
1	1	0	0	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1	1
1	1	1	0	1	1	1	0	1	1
1	1	1	1	1	1	1	1	0	0

Table 21. VHDL Test Results for Emulation Target 7427.

INPUTS (Duplicated)			OUTPUTS		
ER	DR	CR			
EL	DL	CL			
AR	BL	AL	BR	FL	FR
0	0	0	1	1	1
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Table 22. VHDL Test Results for Emulation Target 74S133.

INPUTS													OUT
FR	ER	DR	CR	BR	AR	GL	FL	EL	DL	CL	BL	AL	GR
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0

Appendix K: Test Cell Test Data

for Microcircuit 1.

Table 23. Microcircuit 1, Sample #1, Programming Results with +14 Volts VPP+ and 0 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	7.90	7.00	7.90	7.80	7.00	8.10	8.40	8.30
1	4.00	1.90	1.80	4.50	4.20	2.50	6.70	4.80
2	3.60	1.10	1.30	3.90	3.70	2.00	6.10	4.30
5	2.90	0.50	0.85	3.30	3.00	1.45	5.30	3.50
10	2.50	0.00	0.60	2.70	2.60	1.10	4.70	3.00
20	2.10	-0.30	0.30	2.30	2.30	0.80	4.20	2.30
50	1.60	-0.60	-1.60	1.90	1.70	0.40	3.50	1.50
100	1.30	-0.80	-0.40	1.70	1.40	0.15	3.00	1.00
200	1.00	-1.05	-0.65	1.30	1.00	-0.15	2.50	0.40
500	0.70	-1.40	-0.90	1.00	0.60	-0.50	1.90	-0.20
1000	0.35	-1.60	-1.20	0.80	0.30	-0.70	1.50	-0.70
TIME (mS)								
Chip # 1	VPP+ =	14	VG=	0				
VPP- = 14 V @ VG 8 V and 200 mSec Erase								

Table 24. Microcircuit 1, Sample #1, Programming Results with +14.6 Volts VPP+ and 0 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	8.30	7.10	7.90	7.80	7.40	7.80	8.40	8.40
1	4.00	1.00	1.00	3.55	3.40	1.75	5.50	4.10
2	3.20	0.20	0.70	3.20	2.90	1.20	4.90	4.10
5	2.50	-0.25	0.20	2.80	2.30	0.70	4.10	2.40
10	2.00	-0.50	-0.10	2.50	1.90	0.35	3.50	1.65
20	1.50	-0.80	-0.40	2.10	1.60	0.00	3.00	1.05
50	1.00	-1.00	-0.75	1.70	1.00	-0.40	2.20	0.25
100	0.60	-1.30	-1.05	1.30	0.70	-0.70	1.70	-0.30
200	0.30	-1.60	-1.30	1.00	0.30	-0.90	1.20	-0.80
500	-0.10	-2.10	-1.50	0.55	-0.15	-1.25	0.60	-1.50
1000	-0.40	-2.25	-1.95	0.20	-0.50	-1.50	0.15	-1.90
TIME (mS)								
Chip # 1	VPP+ =	14.6	VG=	0				
VPP- = 14 V @ VG 8 V and 200 mSec Erase								

Table 25. Microcircuit 1, Sample #1, Programming Results with +15.25 Volts VPP+ and +8 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	>8.4	>8.4	>8.4	>8.4	>8.4	>8.4	>8.4	>8.4
1	>8.4	7.15	7.80	>8.4	>8.4	8.10	>8.4	>8.4
2	>8.4	6.80	7.50	>8.4	>8.4	7.70	>8.4	>8.4
5	>8.4	6.60	7.00	>8.4	>8.4	7.35	>8.4	8.35
10	8.35	6.10	6.70	>8.4	8.35	7.00	>8.4	7.80
20	8.20	5.80	6.40	>8.4	8.10	6.80	>8.4	7.30
50	7.80	5.50	6.20	>8.4	7.70	6.40	8.10	6.60
100	7.60	5.30	6.00	8.20	7.40	6.10	7.60	6.10
200	7.20	5.10	5.70	7.80	7.10	5.80	7.20	5.60
500	6.80	4.90	5.50	7.40	6.60	5.55	6.60	5.00
1000	6.50	4.70	5.20	7.10	6.30	5.30	6.30	4.50
TIME (mS)								
Chip # 1	VPP+ =	15.25	VG=	8				
VPP- = 15.4 V @ VG 8 V and 100 mSec Erase								

Table 26. Microcircuit 1, Sample #1, Programming Results with +15.25 Volts VPP+ and 0 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	5.85	5.40	5.45	7.80	5.50	6.50	8.40	7.75
1	2.60	0.05	0.30	1.90	1.60	0.40	3.40	1.70
2	2.60	-0.50	-0.20	1.90	1.35	0.40	2.90	1.35
5	1.90	-1.20	-0.80	1.50	1.20	-0.45	1.95	0.50
10	1.40	-1.65	-1.15	1.10	0.70	-0.80	1.30	-0.10
20	0.95	-1.95	-1.50	0.80	0.30	-1.10	0.70	-0.65
50	0.40	-2.50	-1.90	0.45	-0.20	-1.60	0.00	-1.35
100	0.00	-2.75	-2.20	0.10	-0.50	-1.90	-0.50	-1.85
200	-0.30	-3.00	-2.95	-0.20	-0.90	-2.20	-0.90	-2.35
500	-0.80	-3.15	-2.75	-0.50	-1.30	-2.50	-1.45	-2.90
1000	-1.00	-3.30	-3.00	-0.80	-1.60	-2.75	-1.80	-3.40
TIME (mS)								
Chip # 1	VPP+ =	15.25	VG=	0				
VPP- = 15.4 V @ VG 5 V and 100 mSec Erase								

Table 27. Microcircuit 1, Sample #1, Programming Results with +14.6 Volts VPP+ and +8 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	8.00	7.10	7.10	>8.4	7.00	7.70	>8.4	8.20
1	8.00	6.90	7.05	>8.4	7.00	7.65	>8.4	8.20
2	8.00	6.85	7.05	>8.4	7.00	7.65	>8.4	8.20
5	8.00	6.70	7.00	>8.4	7.00	7.60	>8.4	8.20
10	8.00	6.60	6.90	>8.4	7.00	7.50	>8.4	8.10
20	8.00	6.50	6.80	>8.4	7.00	7.40	>8.4	7.95
50	7.95	6.40	6.70	>8.4	7.00	7.30	>8.4	7.65
100	7.90	6.20	6.50	>8.4	7.00	7.05	>8.4	7.35
200	7.80	6.00	6.30	>8.4	7.00	6.90	>8.4	7.00
500	7.60	5.85	6.10	>8.4	6.95	6.60	>8.4	6.40
1000	7.40	5.70	5.60	8.15	6.90	6.40	8.20	6.00
TIME (mS)								
Chip # 1	VPP+ =	14.6	VG=	8				
VPP- = 14 V @ VG 8 V and 200 mSec Erase								

Table 28. Microcircuit 1, Sample #1, Programming Results with +14 Volts VPP+ and +8 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	8.10	7.00	7.70	>8.4	7.00	7.50	>8.4	8.30
1	8.10	6.90	7.70	>8.4	7.00	7.50	>8.4	8.30
2	8.10	6.90	7.70	>8.4	7.00	7.50	>8.4	8.30
5	8.10	6.90	7.70	>8.4	7.00	7.50	>8.4	8.30
10	8.10	6.80	7.60	>8.4	7.00	7.50	>8.4	8.30
20	8.10	6.80	7.50	>8.4	7.00	7.50	>8.4	8.30
50	8.05	6.70	7.40	>8.4	7.00	7.50	>8.4	8.20
100	8.05	6.60	7.30	>8.4	7.00	7.45	>8.4	8.05
200	8.05	6.50	7.10	>8.4	7.00	7.40	>8.4	7.85
500	8.00	6.40	6.90	>8.4	7.00	7.20	>8.4	7.50
1000	7.90	6.25	6.75	>8.4	7.00	7.10	>8.4	7.20
TIME (mS)								
Chip # 1	VPP+ =	14	VG=	8				
VPP- = 14 V @ VG 8 V and 200 mSec Erase								

Table 29. Microcircuit 1, Sample #3, Programming Results with +12.75 Volts VPP+ and 0 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	>7	>7	>7	>7	>7	>7	>7	>7
1	5.40	6.60	3.80	5.60	2.40	4.20	0.20	>7
2	5.00	6.15	3.30	5.10	1.70	3.70	-0.50	6.70
5	4.60	5.70	2.80	4.60	1.10	3.20	-1.30	6.10
10	4.35	5.20	2.50	4.10	0.85	2.80	-1.80	5.60
20	4.20	5.00	2.10	3.80	0.70	2.40	-2.30	5.10
50	4.00	4.50	1.70	3.40	0.20	1.90	-2.70	4.50
100	3.75	4.30	1.40	3.05	0.00	1.60	-3.10	4.10
200	3.40	4.10	1.15	2.70	-0.30	1.20	-3.40	3.60
500	3.20	3.70	0.70	2.40	-0.70	0.70	-3.70	3.10
1000	2.90	3.40	0.50	2.10	-0.80	0.40	-4.00	2.60
TIME (mS)								
Chip # 3	VPP+ =	12.75	VG=	0				
VPP- = 14.75 V @ VG 7 V and 200 mSec Erase								
VG LIMITED TO +/- 7 V								

Table 30. Microcircuit 1, Sample #3, Programming Results with +12.75 Volts VPP+ and +7 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	>7	>7	>7	>7	>7	>7	>7	>7
1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
10	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
50	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
100	>7	>7	>7	>7	6.60	>7	4.00	>7
200	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
500	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
1000	>7	>7	>7	>7	6.10	>7	3.00	>7
TIME (mS)								
Chip # 3	VPP+ =	12.75	VG=	7				
VPP- = 14.75 V @ VG 7 V and 200 mSec Erase								
VG LIMITED TO +/- 7 V								

Table 31. Microcircuit 1, Sample #4, Programming Results with +14.6 Volts VPP+ and 0 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	7.40	7.60	7.30	8.40	8.30	6.60	7.90	7.80
1	-1.10	0.20	1.90	1.40	-0.80	-0.60	-5.10	-3.30
2	-1.40	-0.25	1.40	0.90	-1.25	-3.10	-5.30	-4.80
5	-2.15	-0.70	0.95	0.30	-1.80	-3.50	-5.30	-5.30
10	-2.40	-0.95	0.65	0.00	-2.10	-3.80	-4.60	-5.70
20	-3.00	-1.70	-0.30	-1.00	-3.20	-4.70	-5.60	-7.00
50	-3.50	-2.15	-0.80	-1.60	-3.80	-5.10	-5.40	-7.55
100	NA	NA	NA	NA	NA	NA	NA	NA
200	NA	NA	NA	NA	NA	NA	NA	NA
500	NA	NA	NA	NA	NA	NA	NA	NA
1000	NA	NA	NA	NA	NA	NA	NA	NA
TIME (mS)								
Chip # 4	VPP+ 14.6		VG=	0				
VPP- = 14 V @ VG 8 V and 200 mSec Erase								

Table 32. Microcircuit 1, Sample #4, Programming Results with +14.6 Volts VPP+ and 0 Volts Control Gate Potential, Second Run.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	7.30	7.60	7.15	8.40	8.30	6.50	8.00	7.70
1	0.40	1.80	3.00	2.25	0.40	-1.40	-3.10	-1.60
2	-0.30	1.00	2.50	1.70	-0.30	-2.15	-4.00	-2.65
5	-0.80	1.90	1.90	1.00	-0.95	-2.75	-4.70	-3.50
10	-1.10	1.55	1.55	0.60	-1.40	-3.10	-5.15	-4.20
20	-1.40	1.20	1.20	0.20	-1.75	-3.40	-5.50	-4.65
50	-1.80	0.75	0.75	-0.30	-2.20	-3.80	-5.90	-5.00
100	-2.10	0.45	0.45	-0.60	-2.55	-4.00	-5.50	-5.40
200	-2.50	1.00	0.10	-0.95	-2.90	-4.20	-5.85	-5.50
500	-2.75	-0.30	-0.30	-1.25	-3.30	-4.70	-6.00	-5.75
1000	-2.90	-0.65	-0.65	-1.45	-3.55	-5.00	-5.80	-6.00
TIME (mS)								
Chip # 4	VPP+ 14.6		VG=	0				
VPP- = 14 V @ VG 8 V and 200 mSec Erase								

Table 33. Microcircuit 1, Sample #4, Programming Results with +14.6 Volts VPP+ and 8 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	7.40	7.70	7.20	8.30	8.30	6.50	8.30	8.00
1	7.20	7.60	7.15	8.30	7.80	5.90	4.70	6.30
2	7.10	7.60	7.10	8.30	7.45	5.60	4.15	5.70
5	7.00	7.50	7.10	8.30	7.00	5.20	3.60	5.40
10	6.80	7.40	7.10	8.10	6.65	4.90	3.20	5.00
20	6.60	7.30	7.00	7.85	6.30	4.50	2.80	4.60
50	6.30	7.15	7.00	7.45	5.90	4.20	2.35	4.20
100	6.00	7.00	7.00	7.20	5.50	3.90	2.00	3.90
200	5.60	6.80	6.90	7.00	5.30	3.70	1.70	3.60
500	5.15	6.50	6.90	6.65	4.90	3.40	1.25	3.10
1000	4.95	6.25	6.80	6.45	4.60	3.10	0.95	2.75
TIME (mS)								
Chip # 4	VPP+ 14.6		VG=	8				
VPP- = 14 V @ VG 8 V and 200 mSec Erase								

Table 34. Microcircuit 1, Sample #4, Programming Results with +14.0 Volts VPP+ and 8 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	7.50	7.90	7.40	8.30	8.30	6.70	8.30	8.10
1	7.40	7.80	7.30	8.30	8.30	6.50	7.00	7.80
2	7.40	7.70	7.30	8.30	8.30	6.30	6.40	7.70
5	7.30	7.70	7.20	8.30	8.20	6.10	5.60	7.30
10	7.20	7.60	7.15	8.30	7.90	6.00	5.20	7.00
20	7.15	7.60	7.15	8.30	7.60	5.80	4.80	6.65
50	7.00	7.50	7.05	8.30	7.10	5.55	4.40	6.25
100	6.80	7.50	7.05	8.30	6.80	5.30	4.00	5.90
200	6.65	7.40	7.00	8.30	6.55	5.10	3.60	5.55
500	6.25	7.30	7.00	8.20	6.20	4.80	3.20	5.20
1000	6.10	7.25	7.00	8.00	5.80	4.50	2.90	4.80
TIME (mS)								
Chip # 4	VPP+ =	14	VG=	8				
VPP- = 14 V @ VG 8 V and 500 mSec Erase								

Table 35. Microcircuit 1, Sample #4, Programming Results with +14.0 Volts VPP+ and 0 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	7.55	7.95	7.46	8.30	6.70	6.75	8.30	8.15
1	1.35	2.60	4.10	3.25	1.75	-0.70	-1.10	0.80
2	1.05	2.18	3.60	2.70	-1.20	-1.20	-1.65	0.25
5	0.30	1.70	3.00	1.90	0.60	-1.75	-2.40	-0.40
10	0.00	1.40	2.60	1.45	0.20	-2.10	-2.90	-0.80
20	-0.65	1.10	2.25	1.10	-0.20	-2.40	-3.30	-1.25
50	-1.10	0.70	1.80	0.70	-0.70	-2.70	-3.80	-1.75
100	-1.45	0.45	1.45	0.45	-1.05	-2.90	-4.20	-2.10
200	-1.60	*	0.20	0.20	-1.35	-3.10	-4.50	-2.40
500	-1.90	-0.20	0.70	-0.10	-1.80	-3.50	-5.00	-2.93
1000	-2.00	-0.50	0.40	-0.30	-2.10	-3.70	-5.10	-3.30
TIME (mS)	* = Data Value Recorded as -1.6V; Probable Test Error.							

Chip # 4	VPP+ =	14	VG=	0	
VPP- = 14 V @ VG 8 V and 500 mSec Erase					

Table 36. Microcircuit 1, Sample #4, Programming Results with +12.75 Volts VPP+ and 0 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	>7	>7	>7	>7	>7	>7	>7	>7
1	2.90	3.80	5.60	4.40	3.60	1.85	2.00	4.50
2	2.40	3.40	5.10	4.00	3.20	1.10	1.40	3.60
5	1.90	3.00	4.60	3.50	2.50	0.60	0.60	3.00
10	1.20	2.80	4.20	3.15	2.10	0.20	0.10	2.60
20	0.90	2.40	3.80	2.80	1.80	0.00	-0.35	2.10
50	0.50	2.10	3.30	2.50	1.30	-0.50	-0.80	1.60
100	0.30	1.90	3.10	2.25	1.00	-0.80	-1.30	1.20
200	0.10	1.65	2.70	2.00	0.60	-1.00	-1.60	0.85
500	0.00	1.40	2.30	1.60	0.20	-1.30	-2.00	0.40
1000	-0.25	1.10	1.90	1.25	-0.10	-1.15	-2.40	0.00
TIME (mS)								
Chip # 4	VPP+ =	12.75	VG=	0				
VPP- = 14.75 V @ VG 7 V and 200 mSec Erase								
VG LIMIT TO +/- 7V								

Table 37. Microcircuit 1, Sample #4, Programming Results with +12.75 Volts VPP+ and 7 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	>7	>7	>7	>7	>7	6.40	>7	>7
1	NA	NA	NA	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA	NA	NA	NA
5	NA	NA	NA	NA	NA	NA	NA	NA
10	NA	NA	NA	NA	NA	NA	NA	NA
20	NA	NA	NA	NA	NA	NA	NA	NA
50	NA	NA	NA	NA	NA	NA	NA	NA
100	>7	>7	>7	>7	>7	5.80	5.80	>7
200	NA	NA	NA	NA	NA	NA	NA	NA
500	6.90	>7	>7	>7	7.00	5.40	5.00	>7
1000	6.70	>7	>7	>7	6.80	5.20	4.70	>7
TIME (mS)								

Chip # 4	VPP+ =	12.75	VG=	7	
VPP- = 14.75 V @ VG 7 V and 200 mSec Erase					
VG LIMIT TO +/- 7V					

Table 38. Microcircuit 1, Sample #6, Programming Results with +12.75 Volts VPP+ and 7 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	>7	>7	>7	>7	>7	>7	>7	>7
1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
10	>7	>7	>7	>7	>7	>7	>7	6.25
20	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
50	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
100	>7	>7	6.60	>7	>7	>7	6.75	5.10
200	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
500	6.90	6.80	6.20	>7	6.40	>7	5.80	4.70
1000	6.70	6.60	5.80	7.00	6.20	>7	5.40	4.00
TIME (mS)								
Chip # 6	VPP+ =	12.75	VG=	7				
VPP- = 14.75 V @ VG 7 V and 200 mSec Erase								
VG LIMITED TO +/- 7 V								

Table 39. Microcircuit 1, Sample #6, Programming Results with +12.75 Volts VPP+ and 0 Volts Control Gate Potential.

AREA	576	256	256	144	64	64	16	16
ID	TCELL2	TCELL3	TCELL4	TCELL5	TCELL7	TCELL8	TCELL9	TCELL10
0	>7	>7	>7	>7	>7	>7	>7	>7
1	2.60	2.60	2.15	3.20	2.30	4.00	3.10	1.50
2	2.15	1.90	1.45	2.75	1.75	3.45	2.40	0.40
5	1.65	1.40	0.95	2.30	1.40	2.90	1.60	-0.35
10	1.40	1.10	0.40	2.00	1.00	2.50	1.00	-0.80
20	1.10	0.80	0.00	1.60	0.70	2.20	0.50	-1.25
50	0.75	0.45	-0.50	1.15	0.30	1.80	0.00	-1.70
100	0.50	0.30	-0.70	0.90	0.00	1.50	-0.50	-2.10
200	0.30	0.00	-0.90	0.70	-0.30	1.20	-0.90	-2.40
500	0.00	-0.25	-1.20	0.30	-0.60	0.80	-1.40	-2.80
1000	-0.25	-0.45	-1.35	0.00	-0.80	0.50	-1.80	-3.10
mS								
Chip # 6	VPP+ =	12.75	VG=	0				
VPP- = 14.75 V @ VG 7 V and 200 mSec Erase								
VG LIMITED TO +/- 7 V								

Appendix L: Microcircuit 2 Test Fixture Schematic Diagram.

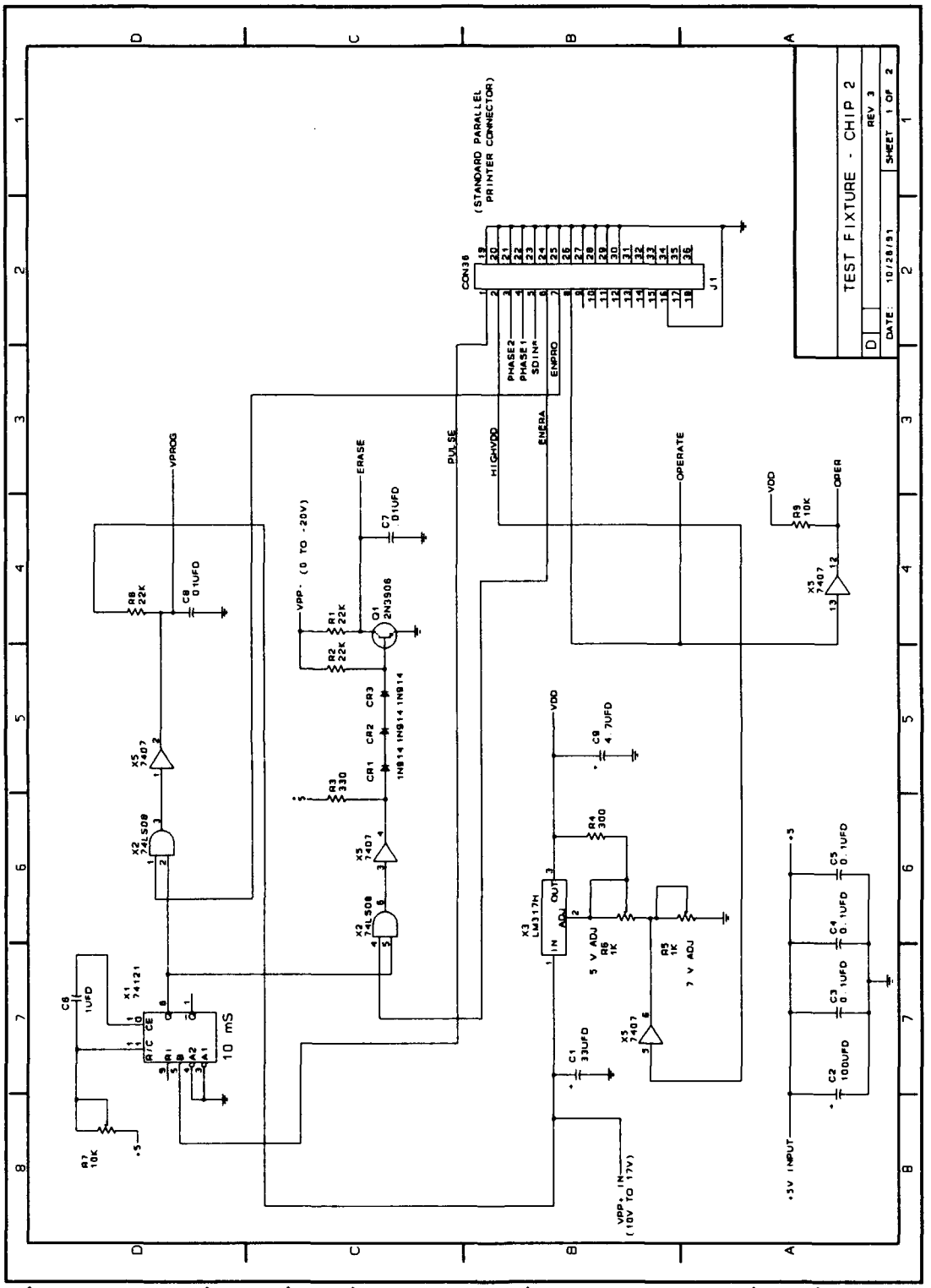


Figure 49. Microcircuit 2 Test Fixture Schematic Diagram. Sheet 1 of 2.

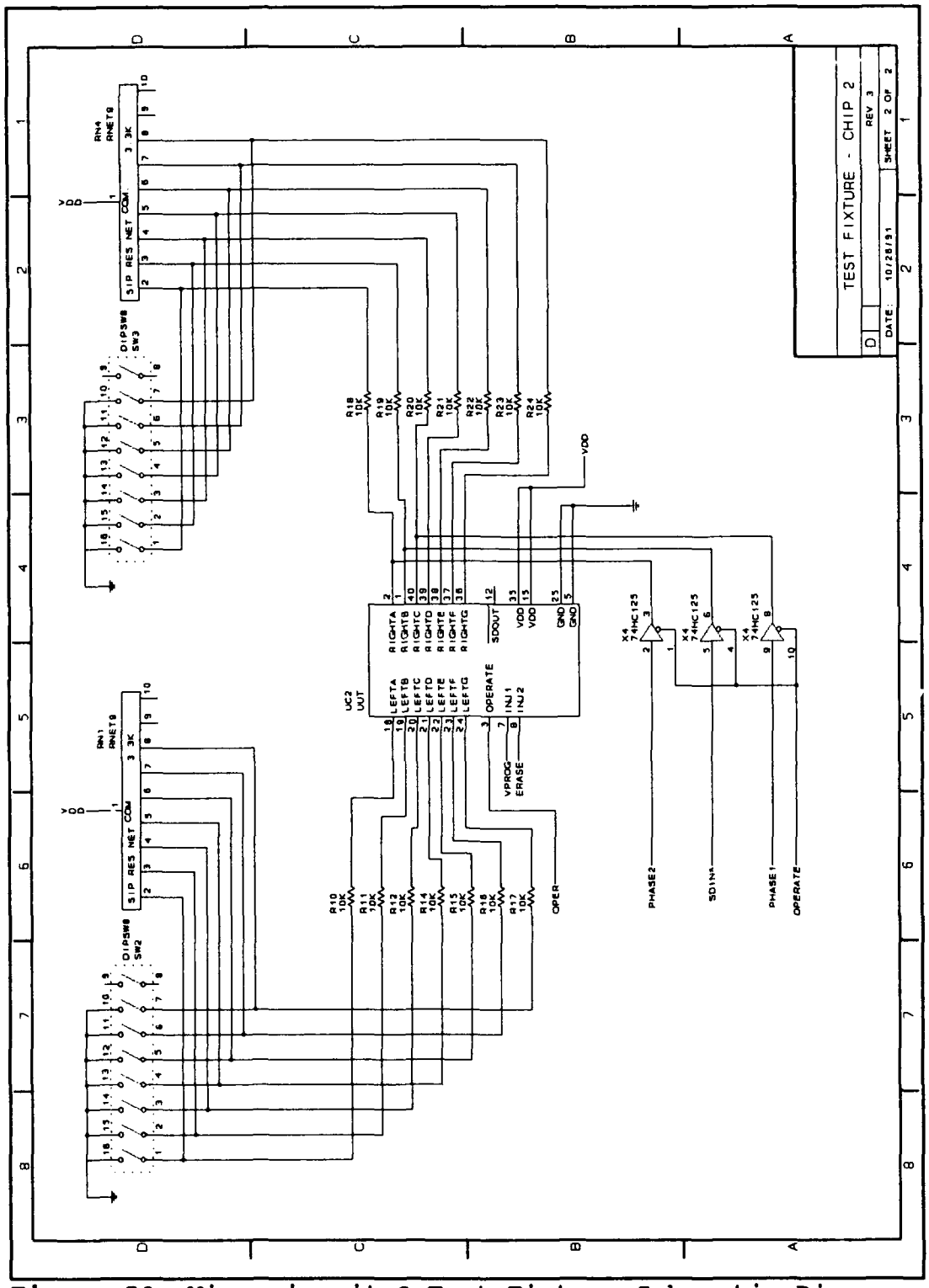


Figure 50. Microcircuit 2 Test Fixture Schematic Diagram. Sheet 2 of 2.

Appendix M: Microcircuit 2 Test Fixture Control Program.

```
program chip2;
{  Program Name:          CHIP2
   Written By:            Joseph V. Breen
   Date Last Modified:    November 1, 1991
   Purpose:               Provides a menu driven interface which
   controls  the test fixture for Microcircuit 2.
   Functions to load the shift register, program the cells,
   and erase all cells are provided.
}
const
{ Masks For Data Bits Of Printer Port Byte }
  highvdd: byte  = 1;
  ph2: byte = 2;
  ph1: byte = 4;
  sdat: byte = 8;
  enera: byte = 16;
  enpro: byte = 32;
  operate:byte = 64;
  addrs: integer = $378;
  stadr: integer = $379;
  pulse_addr: integer = $37a;
  strobeoff: byte = $0a;
  strobeon: byte = $0b;

VAR
  itemp: integer;
  i,CT, ppct, word_count, IPW1, IPW0 : integer;
  temp: BYTE;
  temp2 : byte;
  ch: char;
  istrg: string[20];
  fname: string[20];
  TPW,pw4,pw3,pw2,pw1,pw0 : BYTE;
  Program_Words :text;

procedure putdat(databyte : byte);
begin
   port[addrs] := databyte;
end;

procedure ordat(odata : byte);
VAR
   obyte : byte;
begin
   obyte := port[addrs];
   obyte := obyte or odata;
   port[addrs] := obyte;
end;

procedure anddat(odata : byte);
VAR
   obyte : byte;
begin
   obyte := port[addrs];
   obyte := obyte and not odata;
   port[addrs] := obyte;
end;

Procedure toggl(odata : byte);
begin
     ordat(odata);
     anddat(odata);
end;

FUNCTION HCHAR(inchr:INTEGER) : CHAR;
         begin
         CASE INCHR OF
```

```
          0 : HCHAR := '0';
          1 : HCHAR := '1';
          2 : HCHAR := '2';
          3 : HCHAR := '3';
          4 : HCHAR := '4';
          5 : HCHAR := '5';
          6 : HCHAR := '6';
          7 : HCHAR := '7';
          8 : HCHAR := '8';
          9 : HCHAR := '9';
         10 : HCHAR := 'A';
         11 : HCHAR := 'B';
         12 : HCHAR := 'C';
         13 : HCHAR := 'D';
         14: HCHAR := 'E';
         15 : HCHAR := 'F';
         ELSE WRITELN('HCHAR ERROR !!!!',INCHR);
        end;
end;

procedure writehex(hexchars : byte);
var
tcharh : char;
begin
                tcharh :=HChAR(hexchars div 16);
                write(tcharh);
                tcharh :=HChAR(hexchars mod 16);
                write(tcharh);
end;

Procedure get_integer(var inumb: integer);
var
  icode : integer ;
  instg : string[20];
  tempint : integer;
begin
   icode := 1;
   writeln('');
   repeat
   write('enter number :');
   readln(instg);
   val(instg,tempint,icode);
   if(icode <> 0) then
     begin
       writeln('bad integer. Try again.');
     end;
   until icode = 0;
   write('integer ',tempint,' entered');
   inumb := tempint;
end;

Procedure get_byte(var ibyte: byte);
var
  icode : integer ;
  instg : string[20];
  tempint : integer;
begin
   icode := 1;
   writeln('');
   repeat
   write('enter byte :');
   readln(instg);
   val(instg,tempint,icode);
   if(icode <> 0) then
     begin
       writeln('bad number. Try again.');
     end;
```

```
     if (tempint > 255) then
        begin
          writeln('Maximum exceeded. Maximum is 255 for a Byte value. Try again.');
          icode :=1;
        end;
       until icode = 0;
   ibyte:=tempint;
   write('Byte :  ',ibyte,' entered');
end;

Procedure clok;
begin
    toggl(ph1);
    toggl(ph2);
end;

procedure pulse(pulse_count : integer);
begin
    for CT := 1 to pulse_count do
     begin
         port[pulse_addr] := strobeon;
         delay(1);
         port[pulse_addr] := strobeoff;
         sound(1200);
          delay(15);
         nosound;
    end;
end;

procedure wrprogwd;
         begin
                clok;
                temp := pw0;
                for I := 1 to 8  do
                begin
                   if ((temp mod 2) = 1) then anddat(sdat)
                   else ordat(sdat);
                   clok;
                   temp := temp div 2;
                end;
                temp := pw1;
                for I := 1 to 8  do
                begin
                   if ((temp mod 2) = 1) then anddat(sdat)
                   else ordat(sdat);
                   clok;
                   temp := temp div 2;
                end;
                temp := pw2;
                for I := 1 to 8  do
                begin
                   if ((temp mod 2) = 1) then anddat(sdat)
                   else ordat(sdat);
                   clok;
                   temp := temp div 2;
                end;
                temp := pw3;
                for I := 1 to 8  do
                begin
                   if ((temp mod 2) = 1) then anddat(sdat)
                   else ordat(sdat);
                   clok;
                   temp := temp div 2;
                end;
                temp := pw4;
                for I := 1 to 4  do
                begin
```

```
                if ((temp mod 2) = 1) then anddat(sdat)
                else ordat(sdat);
                clok;
                temp := temp div 2;
              end;
            writeln('');
            write ('Program Word  ');
            writehex(pw4);
            writehex(pw3);
            writehex(pw2);
            writehex(pw1);
            writehex(pw0);
            writeln('  (hex) is written');
end;


Procedure pmenu;
begin;
writeln('            Action Menu For Microcircuit 2 Test Program.');
writeln('');
writeln('     A    Set Operate True');
writeln('     B    Set Operate False');
writeln('     C    Clock Multiple Times (Prompt For Number)');
writeln('     D    Default Signals (All zeros)');
writeln('     E    Set ENERA true and Program false');
writeln('     F    Set Program True, ENERA False.');
writeln('     L    Load Byte (8 bits) into Serial Register (Prompt For Byte)');
writeln('     M    Load Lower n bits of Byte into Serial Register (Prompt for Byte & #) ');
writeln('     P    Pulse One Time (VDD+ or VDD-  pulse if ENPRO or ENERA true');
writeln('     Q    Quit Program');
writeln('     R    Reset Serial Input Data to 0');
writeln('     S    Set Serial Input Data to 1');
writeln('     T    Single Tick of Phase 1 Then Phase 2 Clocks');
writeln('     U    ERASE (PH1 & 2 true, Input = 1, 20 VPP- pulses');
writeln(' W    Load Program Word         X    Get and Store Prog. Word');
writeln(' Y    Set Ph1 & Ph2 False(Dflt) Z    Set Ph1 & Ph2 True');
writeln(' 0    Set HIGHVDD True          1    Set HIGHVDD False');
writeln(' 2    Get Filename              3    Set Program Pulse Count');
writeln(' 4    Program Pulses            5    Program With File Data');
end;


  begin
    putdat(0);
    ppct := 10;
WRITELN;
WRITELN('         TEST PROGRAM FOR MICROCIRCUIT 2    ');
WRITELN('       November 1, 1991         V1.2       ');
WRITELN('       Written By:   J. Breen, AFIT EN     ');
WRITELN;

begin
{Make highvdd false, both clocks  and data false}
   putdat(0);
   pmenu;

   repeat
      read(kbd,ch);
      ch := upcase(ch);
      write(ch);
      case ch of {main action loop}
        'A' : begin
                ordat(operate);
              end;

        'B' : begin
```

```
        anddat(operate);
      end;

'C' : begin
        get_integer(itemp);
        writeln(' Clocking ',itemp, '  times.');
        for I := 1 to itemp do clok;
        writeln('Done.');
      end;

'D' : begin
        putdat(0);    {    Write to port }
      end;

'E' : begin
        anddat(enpro);
        ordat(enera);
      end;

'F' : begin
        anddat(enera);
        ordat(enpro);
      end;

'L' : begin
        get_byte(temp);
        writeln(' Byte Value  ',temp, ' accepted.');
        for I := 1 to 8  do
        begin
           if ((temp mod 2) = 1) then ordat(sdat)
           else anddat(sdat);
           clok;
           writeln(temp mod 2);
           temp := temp div 2;
        end;
        writeln('Done.');
      end;

'M' : begin
        writeln('');
        writeln('How many bits of data do you want to load (0 - 8)?');
        get_byte(temp);
        writeln('');
        if temp <> 0 then
        begin
        temp := temp - 1;
        temp := temp mod 8;
        temp2 := temp;
        writeln('What is the data byte to be used?');
        get_byte(temp);
        writeln(' Byte Value  ',temp, ' accepted.');
        for I := 0 to temp2  do
        begin
           if ((temp mod 2) = 1) then ordat(sdat)
           else anddat(sdat);
           clok;
           writeln(temp mod 2);
           temp := temp div 2;
        end;
        end;
        writeln('Done.');
      end;

'P' : begin
       i := 1;
       pulse(i);
      end;
```

```
    'Q' :;  { do nothing - going to quit}

    'R' : begin
            ordat(sdat); {SDAT is inverted SDIN}
          end;

    'S' : begin
            anddat(sdat); {Inverted SDIN}
          end;

    'T' : begin
            clok;
          end;

    'U' : begin
            putdat(0);
            ordat(enera);
            ordat(ph1);
            ordat(ph2);
            ordat(sdat);
{SDAT is 1, SDIN is 0, all cgates are 1}
            delay(1);
            pulse(20);
            putdat(0);
            ordat(operate);
          end;


    'W' : begin
            wrprogwd;
          end;

    'X' : begin
          writeln(' Use $ for Hexadecimal entry  ');
          writeln(' Enter Most Sig. Byte of Program Word.');
          get_byte(pw4);
          writeln(' Enter Second Byte of Program Word.');
          get_byte(pw3);
          writeln(' Enter Third Byte of Program Word.');
          get_byte(pw2);
          writeln(' Enter Fourth Byte of Program Word.');
          get_byte(pw1);
          writeln(' Enter Least Sig. Byte of Program Word.');
          get_byte(pw0);
          writeln('');
          write ('Program Word  ');
          writehex(pw4);
          writehex(pw3);
          writehex(pw2);
          writehex(pw1);
          writehex(pw0);
          writeln('  (hex) is entered');
         end;

    'Y' : begin
            anddat(ph1);
            anddat(ph2);
          end;

    'Z' : begin
            ordat(ph1);
            ordat(ph2);
          end;

    'O' : begin
            ordat(highvdd);
          end;
```

```
'1' : begin
        anddat(highvdd);
      end;

'2' : begin
      Writeln('');
      Write('enter file name: ');
      read(fname);
      assign(program_words,fname);
      reset(program_words);
      writeln('');
      writeln('File ',fname, ' is selected');
     end;

'3' : begin
       writeln(' Enter Number of Program Pulses to Use');
       get_integer(ppct);
       Writeln('');
       writeln( ppct, ' pulses will be used for programming');
     end;

'4' : begin
       ordat(highvdd);
       ordat(enpro);
       pulse(ppct);
       anddat(highvdd);
       anddat(enpro);
     end;

'5' : begin
     putdat(0);
     reset(program_words);
     read(program_words,word_count);
     writeln('');
     writeln(word_count, '  Program Words');
     for CT := 1 to word_count do
     begin
       readln(program_words);
       read(program_Words,IPW1,IPW0);
       Writeln('');
       Writeln('GPW = ',IPW1, '  Select = ',IPW0);
       PW4 := IPW1 shr 4;
       PW3 := 0;
       PW2 := 0;
       PW1 := 0;
       PW0 := 0;
       TPW := 1;
       case IPW0 of
         0..7: PW0 := TPW SHL IPW0;
         8..15: PW1 := TPW SHL (IPW0 - 8);
         16..23: PW2 := TPW SHL (IPW0 - 16);
         24..27: PW3 := TPW SHL (IPW0 - 24);
         ELSE  Writeln('Select out of range.');
       end;
       PW3 := PW3 OR ((IPW1 shl 4) and $ff);
       wrprogwd;
       ordat(highvdd);
       ordat(enpro);
       pulse(ppct);
       anddat(highvdd);
       anddat(enpro);
  end;
      putdat(0);
      ordat(operate);  (set operate true)
  end;

else begin
```

```
            sound(510);
            delay(300);
            nosound;
            writeln('');
            writeln(' Unrecognized input ',ch);
            pmenu;
          end;
      end;

      until ((ch = 'q') or (ch = 'Q'));
    end;
end.
```

References

1. D. H. Thornhill. "The Microcircuit Emulation Program: A New Solution to the Discontinued Parts Problem," *GOMAC (Government Microcircuits Applications Conference) Digest Of Papers Held in Orlando, Florida on 27-29 October 1987,* pp. 371-374, Commanding General, U.S. Army LABCOM, Attn: SLCET-DT, Fort Monmouth, NJ 07703, October 1987 (AD-B119-187).

2. "DOD Seeks New Sources for Old Semiconductors," *Electronics,* November 11, 1985, pp. 18-19.

3. A. Daudelin and R. Ferra. "Re-Engineering of Discontinued Parts: Two Case Studies," *GOMAC (Government Microcircuits Applications Conference) Digest Of Papers Held in Orlando, Florida on 27-29 October 1987,* pp. 367-370, Commanding General, U.S. Army LABCOM, Attn: SLCET-DT, Fort Monmouth, NJ 07703, October 1987 (AD-B119-187).

4. G. Leopold. "Shortage of Obsolete Chips Makes it Tough on Military," *Electronics*, October 14, 1985, pp. 42-43.

5. R. King. *A Replacement Strategy for Obsolete Integrated Circuits: Final Technical Report*, N00164-87-C0207, Naval Weapons Support Center, Crane Indiana 47522-5011, March 1988 (AD-B123 853).

6. G. Gaugler, T. Glum, and G. Wheeler. "A Bipolar Device Array for the Replacement of Obsolete IC Families," *GOMAC (Government Microcircuits Applications Conference) Digest Of Papers Held in Orlando, Florida on 27-29 October 1987,* pp. 359-361, Commanding General, U.S. Army LABCOM, Attn: SLCET-DT, Fort Monmouth, NJ 07703, October 1987 (AD-B119-187).

7. C. R. Regan, T.A. Shull, and R.M. Holloway. "Investigation Into Replacement of Discrete ICs with Programmable Logic Devices in a NASA Flight Instrument Subsystem," *GOMAC (Government Microcircuits Applications Conference) Digest Of Papers Held in Orlando, Florida on 27-29 October 1987,* pp. 364-366, Commanding General, U.S. Army LABCOM, Attn: SLCET-DT, Fort Monmouth, NJ 07703, October 1987 (AD-B119-187).

8. L. R. Carley. "Trimming Analog Circuits Using Floating-Gate MOS Memory," *IEEE Journal of Solid-State Circuits,* vol. 24, no. 6, pp. 1569-1575, December 1989.

9. E. H. Snow. "Fowler-Nordheim Tunneling in SiO_2 Films," *Solid State Communications,* 1967, vol. 5, pp. 813-815.

10. M. Lenzlinger and E. H. Snow. "Fowler-Nordheim Tunneling into Thermally Grown SiO_2," *Journal of Applied Physics,* vol. 40, no. 1, pp. 278-283, Jan. 1969.

11. Z. A. Weinberg. "On Tunneling in Metal-Oxide-Silicon Structures," *Journal of Applied Physics*, vol. 53, no. 7, pp. 5052-5056, July 1982.

12. A. Kolodny, S. T. K. Nieh, B. Eitan, and J. Shappir. "Analysis and Modeling of Floating-Gate EEPROM Cells," *IEEE Trans. Electron Devices.*, vol. ED-33, pp. 835-843, June 1986.

13. D. Kahng and S. M. Sze. "A Floating Gate and Its Application to Memory Devices," *The Bell System Technical Journal*, vol. 46, no. 6, pp. 1288-1295, July 1967.

14. D. Frohman-Bentchkowsky. "FAMOS - A New Semiconductor Charge Storage Device," *Solid State Electronics,* 1974, vol. 17, pp. 517-529.

15. J. R. Mann. "Floating Gate Circuits in MOSIS," Technical Report 824, Massachusetts Institute of Technology Lincoln Laboratory, Nov. 1, 1990.

16. M. Momodomi *et al.*, "An Experimental 4-Mbit CMOS EEPROM with a NAND Structured Cell," *IEEE Journal of Solid-State Circuits,* vol. 24 no. 5, pp. 1238-1243, October 1989.

17. C. Bleiker and H. Melchior. "A Four-State EEPROM Using Floating-Gate Memory Cells," *IEEE Journal of Solid-State Circuits,* vol. SC-22, no. 3, pp. 460-463, June 1987.

18. A. Thomsen and M. A. Brooke, "A Floating-Gate MOSFET with Tunneling Injector Fabricated Using a Standard Double-Polysilicon CMOS Process," *IEEE Electron Device Letters,* vol. 12, no. 3, pp. 111-113, March 1991.